

Appendix F: Testing System

TABLE OF CONTENTS

1 Background.....	1-1
2 Preliminary Requirements	2-1
2.1 Introduction	2-1
2.2 Development Requirements.....	2-1
2.3 Applications.....	2-2
2.3.1 On-Board Unit (OBU) Test Application	2-2
2.3.2 OBU Test and RSU Test Common Requirements	2-4
3 Application Message Specification	3-1
3.1 Introduction	3-1
3.1.1 Scope.....	3-1
3.1.2 Application Interfaces	3-1
3.1.3 GPS Receiver Interface	3-2
3.1.4 Vehicle Bus Interface	3-2
3.1.5 Traffic Signal Interface	3-2
3.1.6 WAVE Radio Module Interface.....	3-2
3.2 Message Definitions	3-3
3.2.1 Common Message Header.....	3-3
3.2.2 OBU V2V Safety Message.....	3-4
3.2.3 RSU Traffic Signal Message.....	3-6
4 Testing System Specification and Architecture	4-1
4.1 Introduction	4-1
4.1.1 Scope.....	4-1
4.2 Architecture	4-1
4.2.1 Overview	4-1
4.2.2 External Interfaces.....	4-1
4.2.3 User Interface	4-2
4.2.4 Test Control.....	4-4

4.3	Class Descriptions	4-6
4.3.1	Class Descriptions	4-7
4.4	Data	4-25
4.4.1	Structures.....	4-25
4.4.2	Global Variables.....	4-27
4.5	Design Goals and Constraints.....	4-29
4.5.1	Design Methodology	4-29
4.5.2	MFC, Standard Library Usage	4-29
4.5.3	ANSI Compliance	4-29
4.5.4	Memory Management	4-29
4.5.5	Naming Conventions.....	4-29
4.6	Modifications and Enhancements	4-29
4.6.1	Changing Over the Air Formats	4-29
4.6.2	Changing Data Logging	4-30
4.6.3	Changing Test Types.....	4-30
4.6.4	Changing T9App Configuration Loading / Saving	4-30
4.6.5	Changing CAN messages	4-31
5	Validation Results	5-1
5.1	Introduction	5-1
5.1.1	Scope	5-1
5.2	Test Configurations	5-1
5.2.1	WAVE Radio Module Network Connection.....	5-1
5.2.2	Test Setups	5-1
5.2.3	Initialization	5-3
5.2.4	Software Builds	5-6
5.3	GUI Parameter Tests.....	5-6
5.3.1	WRM Configuration Screen.....	5-6
5.3.2	Comm Parameters Screen	5-9
5.3.3	Test Options Screen	5-14
5.3.4	Traffic Signal Information Screen.....	5-17
5.3.5	Test Display Screen.....	5-20
5.3.6	GUI Configuration Recording, Persistence.....	5-25

5.4	Interface Tests.....	5-26
5.4.1	GPS Receiver Interface Tests.....	5-26
5.4.2	Vehicle Bus Interface Tests.....	5-30
5.4.3	Traffic Signal Interface Tests.....	5-39
5.5	Requirement/Test Cross Reference Matrix.....	5-45
5.5.1	Task 9 Software Requirement Verification.....	5-45
5.5.2	Common Message Header Verification	5-48
5.5.3	OBU Message Verification	5-48
5.5.4	RSU Message Verification.....	5-49
6	Appendix	6-1
6.1	Terms, Acronyms, and Abbreviations	6-1
6.2	References	6-3
6.3	Task 9 Device CAN Input Messages.....	6-3
6.3.1	Vehicle Velocity Message.....	6-3
6.3.2	Vehicle Acceleration Message	6-4
6.3.3	Vehicle Devices Message.....	6-5
6.4	Traffic Signal Interface.....	6-5
6.4.1	Initial Request/Response State	6-6
6.4.2	Periodic Request/Response State	6-6
6.5	Preliminary Vehicle-to-Vehicle Common Message Set.....	6-9
6.6	Task 9 Application Users' Guide	6-11
6.6.1	Introduction	6-11
6.6.2	Scope	6-11
6.6.3	Setup Procedures	6-11
6.6.4	Operating Instructions	6-14
6.6.5	Troubleshooting	6-25

TABLE OF FIGURES

Figure 2-1. Task 9 Device in the OBU Test Application Configuration.....	2-2
Figure 2-2. Task 9 Device in the RSU Test Application Configuration	2-3
Figure 3-1. OBU Interfaces	3-1
Figure 3-2. RSU Interfaces.....	3-2
Figure 3-3. Common Message Header Format	3-3
Figure 3-4. OBU V2V Safety Message Format	3-5
Figure 3-5. RSU Traffic Signal Message Format (Part 1 of 2)	3-7
Figure 3-6. RSU Traffic Signal Message Format (Part 2 of 2)	3-8
Figure 4-1. T9App Architecture.....	4-1
Figure 4-2. T9App External Interfaces.....	4-2
Figure 4-3. Configuration Dialogs Screen Shots.....	4-3
Figure 4-4. Test Display Dialog Screen Shot.....	4-3
Figure 4-5. T9App Test Execution Architecture (UML Diagram).....	4-5
Figure 5-1. Generic Test Setup.....	5-2
Figure 5-2. OBU Test Setup.....	5-2
Figure 5-3. RSU Test Setup	5-3
Figure 6-1. Common Setup	6-11
Figure 6-2. OBU Setup.....	6-13
Figure 6-3. RSU Setup	6-14
Figure 6-4. T9App Main Screen Layout	6-15
Figure 6-5. WRM Configuration Screen Layout.....	6-15
Figure 6-6. Comm Parameters Screen Layout.....	6-16
Figure 6-7. Test Options Screen Layout.....	6-18
Figure 6-8. Traffic Signal Information Screen Layout.....	6-19
Figure 6-9. Location Information Screen Dialog Box.....	6-19
Figure 6-10. Test Display Screen Layout.....	6-20
Figure 6-11. Sample OBU Logfile	6-23
Figure 6-12. Excel Text Import Wizard	6-24
Figure 6-13. Worksheet Selection	6-24
Figure 6-14. Imported Logfile.....	6-25

Figure 6-15. COM Port Conflict Error Message	6-26
Figure 6-16. GPS Port Initialization Error Message.....	6-26
Figure 6-17. Traffic Signal Port Initialization Error Message.....	6-27
Figure 6-18. CAN Bus Initialization Error Message Box	6-27

TABLE OF TABLES

Table 2-1. Definitions Used Throughout the Document	2-1
Table 3-1. Common Message Header Field Descriptions	3-4
Table 3-2. OBU V2V Safety Message Discrete Field Definitions	3-6
Table 3-3. RSU Traffic Signal Message Discrete Field Definitions	3-9
Table 4-1. T9App Dialog Class Summary	4-4
Table 4-2. T9App Class Summary	4-6
Table 4-3. CCanControl Methods and Attributes.....	4-8
Table 4-4. CCanMsg Methods and Attributes.....	4-8
Table 4-5. CCommParams Methods and Attributes.....	4-9
Table 4-6. CDeviceDatabase Methods and Attributes	4-10
Table 4-7. CGpsControl Methods and Attributes.....	4-11
Table 4-8. CGpsPosition Methods and Attributes.....	4-11
Table 4-9. CLocationInfo Methods and Attributes	4-12
Table 4-10. CSerialPort Class Methods and Attributes.....	4-15
Table 4-11. CSignalControl Methods and Attributes	4-15
Table 4-12. CSignalControl State Machine Events.....	4-16
Table 4-13. CSignalInfo Methods and Attributes	4-17
Table 4-14. CTestLog Methods and Attributes	4-18
Table 4-15. CTestStatusDialog Methods and Attributes.....	4-18
Table 4-16. CTestStatusDialog Processes	4-19
Table 4-17. CVehicleInfo Methods and Attributes	4-20
Table 4-18. CWaveDeviceInfo Methods and Attributes	4-22
Table 4-19. CWrmControl Methods and Attributes	4-23
Table 4-20. CWrmMsg Methods and Attributes	4-24
Table 4-21. GPS Location Structure	4-25

Table 4-22. Test Header Structure.....	4-25
Table 4-23. Stopping Location Structure	4-26
Table 4-24. Common Message Structure	4-26
Table 4-25. OBU VTV Message	4-26
Table 4-26. RSU TS Message	4-27
Table 4-27. T9 App Globals.....	4-27
Table 4-28. WAVE API Globals.....	4-28
Table 4-29. PCAN USB Globals.....	4-28
Table 5-1. OBU Initialization Procedure.....	5-3
Table 5-2. RSU Initialization Procedure	5-4
Table 5-3. Communication Parameters Initial Values.....	5-4
Table 5-4. Test Options Initial Values	5-5
Table 5-5. Traffic Signal Parameters Initial Values	5-5
Table 5-6. WRM and T9App Software Versions	5-6
Table 5-7. WRM Configuration Screen Results.....	5-7
Table 5-8. WRM Configuration Data Elements	5-9
Table 5-9. Comm Parameters Screen Low Parameter Values.....	5-10
Table 5-10. Comm Parameters Screen Low Parameter Results	5-10
Table 5-11. Comm Parameters Screen Data Elements.....	5-11
Table 5-12. Comm Parameters Screen Mid Parameter Values	5-11
Table 5-13. Comm Parameters Screen Mid Parameter Results.....	5-12
Table 5-14. Comm Parameters Screen High Parameter Values	5-13
Table 5-15. Comm Parameters Screen High Parameter Results	5-13
Table 5-18. Test Options Screen Parameter Values	5-14
Table 5-19. Test Options Screen Case 1 Results.....	5-15
Table 5-20. Test Options Data Elements.....	5-16
Table 5-21. Test Options Screen Case 2 Results.....	5-16
Table 5-22. Test Options Screen Case 3 Results.....	5-17
Table 5-23. Traffic Signal Information Screen Results.....	5-18
Table 5-24. Traffic Signal Message Data Elements	5-19
Table 5-25. Distance, Heading Test GPS Data	5-21
Table 5-26. Distance, Heading Test Parameter Values	5-21
Table 5-27. Distance, Heading Test Results.....	5-21

Table 5-28. RSU – RSU Bearing Test GPS Data.....	5-22
Table 5-29. RSU – RSU Bearing Test Parameter Values	5-22
Table 5-30. RSU – RSU Bearing Test Results.....	5-23
Table 5-31. OBU – OBU Relative Heading Test Parameters	5-24
Table 5-32. OBU – OBU Relative Heading Test Results	5-25
Table 5-33. GUI Configuration Recording, Persistence Results	5-26
Table 5-34. Common Message Header GPS Data Test Results	5-27
Table 5-35. Common Message Header GPS Data Elements.....	5-27
Table 5-36. OBU GPS Interface Test Results	5-28
Table 5-37. OBU V2V Safety Message GPS Data Elements.....	5-28
Table 5-38. RSU GPS Interface Test Results.....	5-30
Table 5-39. RSU Traffic Signal Message GPS Data Elements	5-30
Table 5-40. Vehicle Velocity Settings (Low-Value).....	5-31
Table 5-41. Vehicle Acceleration Settings (Low-Value)	5-31
Table 5-42. Vehicle Devices Settings (Low-Value).....	5-32
Table 5-43. Vehicle Interface Low-Value Test Results	5-32
Table 5-44. OBU V2V Safety Message Vehicle Data Elements.....	5-33
Table 5-45. Vehicle Velocity Settings (Mid-Value)	5-34
Table 5-46. Vehicle Acceleration Settings (Mid-Value).....	5-34
Table 5-47. Vehicle Devices Settings (Mid-Value)	5-34
Table 5-48. Vehicle Interface Mid-Value Test Results	5-35
Table 5-49. Vehicle Velocity Settings (High-Value)	5-35
Table 5-50. Vehicle Acceleration Settings (High-Value).....	5-36
Table 5-51. Vehicle Devices Settings (High-Value)	5-36
Table 5-52. Vehicle Interface High-Value Test Results.....	5-36
Table 5-53. Vehicle Devices Settings	5-37
Table 5-54. Vehicle Interface Test Results	5-37
Table 5-56. Vehicle Devices Settings	5-38
Table 5-57. Vehicle Interface Test Results	5-38
Table 5-58. Vehicle Devices Settings	5-38
Table 5-60. Vehicle Interface Test Results	5-38
Table 5-61. Traffic Signal Long Initial Response, Fast Polling Results	5-39
Table 5-62. Short Initial Response, Slow Polling Results.....	5-40

Table 5-63. Traffic Signal Active Phase 1 and 5 Parameters	5-41
Table 5-64. Traffic Signal Active Phase 1 and 5 Results	5-41
Table 5-65. RSU Traffic Signal Message Data Elements	5-41
Table 5-66. Traffic Signal Active Phase 2 and 6 Parameters	5-42
Table 5-67. Traffic Signal Active Phase 2 and 6 Results	5-42
Table 5-69. Traffic Signal Active Phase 3 and 7 Parameters	5-43
Table 5-70. Traffic Signal Active Phase 3 and 7 Results	5-43
Table 5-71. Traffic Signal Active Phase 4 and 8 Parameters	5-44
Table 5-72. Traffic Signal Active Phase 4 and 8 Results	5-44
Table 5-73. Task 9 Software Requirement Cross Reference.....	5-45
Table 5-74. Common Message Header Field Verification.....	5-48
Table 5-75. OBU Message Verification	5-48
Table 5-76. RSU Message Verification.....	5-49
Table 6-1. Comm Parameters Screen Field Definitions	6-16
Table 6-2. Test Option Screen Field Definitions.....	6-18
Table 6-3. Traffic Signal Information Screen Field Definitions	6-19
Table 6-4. Test Display Screen Field Definitions	6-21
Table 6-5. Logfile Record Types.....	6-22

1 Background

The WAVE radio modules developed under Task 6D include the capability for hardware and software operation at 5.9 GHz, closely compliant with available lower layer DSRC standards. The WAVE radio modules also include a well-documented Applications Programming Interface (API) that includes access to many of the functions available in the DSRC lower layers. The development of a common computing platform was necessary to support future VSCC testing and evaluation of functionalities being specified in the upper layer standards. This common computing platform was also required for simulating transmissions from traffic signal controllers, as well as interfacing with these controllers to transmit traffic signal data in real time.

The next generation testing system developed under Task 9 interfaces with automotive network systems, traffic signal controllers and research computers. As well, the system supports enhanced data acquisition capabilities, both for recording pertinent test-related information, and for acquiring and recording automotive network data pertinent to the testing in Task 10. Due to the wide range of anticipated testing uses, the next generation test system was designed to function as a portable system with flexible system interfaces to plug into relevant automotive networks, traffic signal controllers and research computers, as required.

The preliminary testing of communications functionalities necessary for the development of prototype vehicle safety applications depended upon access to proprietary in-vehicle networks and systems, as well as access to the DSRC communications capabilities. The API on the WAVE radio modules was also designed to be available for direct access by the proprietary on-board computing systems of VSCC member companies. Development of software on these proprietary systems to support this data integration and functionality testing represented an additional development activity that was necessary to enable Task 10 testing and evaluation.

The contents of Chapters 3-5 and portions of Chapter 6 of this report were prepared by the identified contractor, Denso LA Laboratories, in conjunction with the VSCC. The materials prepared by Denso comprised a number of separate specifications and reports. These were integrated into Chapters 3-6 of this report with minor editorial and formatting revisions.

This project was completed successfully within the necessary short time constraints imposed by the Task 10 testing requirements. The Task 9 software was delivered in time, and in sufficient working order, to support the final phases of the Task 10 testing, including real time vehicle bus data transfer between vehicles of different manufacturers, as well as real time traffic controller data transmission from intersection locations to vehicles in motion.

2 Preliminary Requirements

The preliminary requirements presented in this chapter represent the starting point for the development of the Task 9 software, thus are represented in the future tense. These preliminary requirements were subsequently refined under an iterative process between Denso and the VSCC in order to complete the detailed specifications described later in this report.

2.1 Introduction

The Task 9 test software and Task 9 device will be used to support testing for VSC task 10. The task 10 testing will involve running scenario specific tests involving two or more vehicles and collecting data for analysis of each test.

The primary roles of the Task 9 device shall be to:

- Interface between a vehicle communication bus or a roadside device and the Wave Radio Module (WRM) so that the WRM can transmit information from a vehicle or roadside device to neighboring vehicles.
- Receive and decode incoming messages from the WRM transmitted from surrounding vehicles or roadside devices.
- Record information sent to and information received from other devices through the WRM.
- Allow an interface to change WRM parameters and application-specific communication parameters.

Table 2-1 refers to commonly used terms and their definitions.

Table 2-1. Definitions Used Throughout the Document

Test Scenario Parameters	communication parameters to be set according to the test scenario defined in task 10
Wave Radio Module (WRM)	configurable radio module provided as a work output from VSC Task 6D

2.2 Development Requirements

The Task 9 device shall be the existing laptop computers currently used as part of the VSC communications test kit, with Windows operating system installed. The software to be developed by the supplier for VSC Task 9 shall be configured to operate on the Task 9 device. The deliverables for Task 9 shall include validation of correct functional operation of developed software on the Task 9 device, plus all source code, development environments and any necessary licensing so that members of the VSC consortium can freely modify the source code to perform specific tasks and create new applications.

2.3 Applications

The Task 9 software shall have the capability to run one application at a time with the Task 9 device being configurable to easily choose amongst a set of preloaded applications.

As part of this task, the supplier developing the Task 9 software shall provide two basic applications for use on the Task 9 device. The on-board unit (OBU) test application will be used in a vehicle and will wirelessly broadcast vehicle parameters and decode incoming packets containing surrounding vehicle parameters. The roadside unit (RSU) test application will be used near a roadway infrastructure device such as a traffic signal and will broadcast traffic signal information. The next two subsections discuss these two applications and the unique requirements of each. The last subsection discusses common requirements of these applications.

2.3.1 On-Board Unit (OBU) Test Application

Figure 2-1 shows the basic physical connections the OBU Test application will utilize.

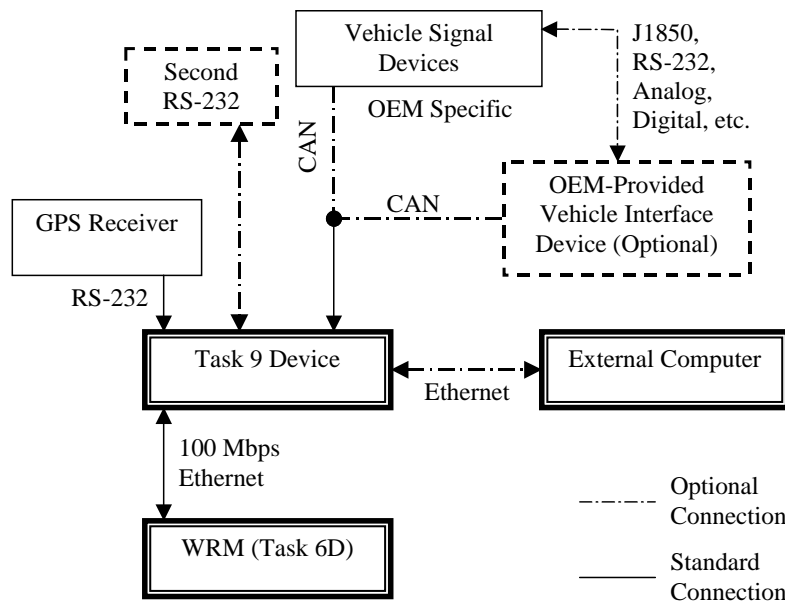


Figure 2-1. Task 9 Device in the OBU Test Application Configuration

2.3.1.1 OEM Vehicle Interface

In the OBU test application, the Task 9 device shall use a Controller Area Network (CAN) bus to receive messages with vehicle information such as vehicle speed, brake position, acceleration, etc. The format of the information received on the CAN bus is defined in Section 6.3.

The OEM Vehicle Interface software shall be written so that OEMs may modify the source code to achieve an alternative mapping from CAN message identifiers and formats to vehicle data.

2.3.1.2 Wireless Data Output

The OBU test application shall periodically issue commands to the WRM to transmit the latest GPS and vehicle information wirelessly. The periodicity of the transmitted information shall be adjustable through the control user interface and shall allow periods at least as small as 10 milliseconds. The format of the data sent wirelessly is defined in Section 3.2.2.

2.3.1.3 Roadside Unit (RSU) Test Application

Figure 2-2 shows the basic physical connections the RSU test application will utilize.

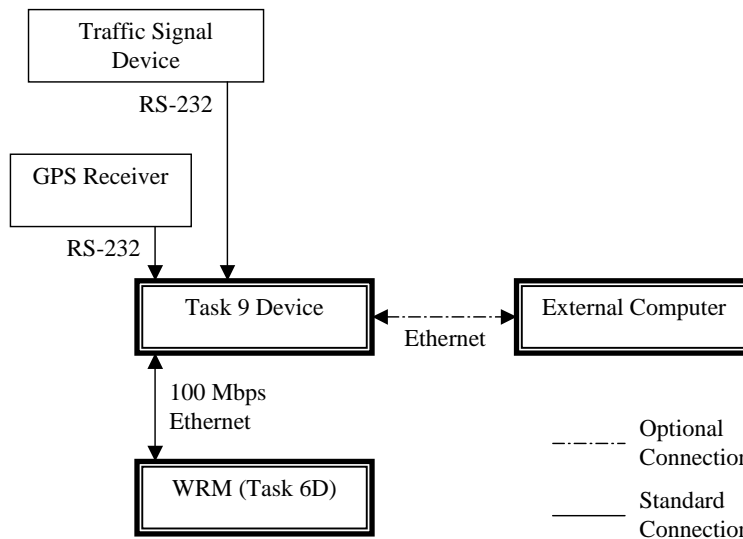


Figure 2-2. Task 9 Device in the RSU Test Application Configuration

2.3.1.4 Traffic Signal Device Interface

The RSU test application shall decode traffic signal information from an RS-232 connection. The format of the information contained on the RS-232 connection is defined in Section 6.4.

2.3.1.5 Wireless Data Output

The RSU test application shall periodically issue commands to the WRM to transmit traffic signal information wirelessly. The periodicity of the transmitted information shall be adjustable through the control user interface and shall allow periods at least as small as 10 milliseconds.

The format and content of the data sent wirelessly, including GPS and traffic signal information from the RS-232 connection on the Task 9 device, is defined in Section 3.2.3.

2.3.1.6 Application Display

Control User Interface

The RSU test application shall be configurable so that the user has the capability to modify constant parameters for the traffic signal information. Example constant parameters that may require adjustment via the control user interface include location of the intersection, directionality, stopping locations, number of lanes, etc. The constant parameters that require adjustment will be defined by the members of the VSC consortium.

2.3.2 OBU Test and RSU Test Common Requirements

2.3.2.1 GPS Receiver Interface

In the OBU test application, the Task 9 device shall use an RS-232 port to receive ASCII messages formatted in the National Maritime Electronics Association (NMEA) 0183 standard. At a minimum, the GPS receiver will be configured to output the "GPGGA" and "GPVTG" strings defined by NMEA 0183. The application shall parse these messages to get the GPS data required in other parts of this document.

2.3.2.2 Interface to WRM

Telnet Interface to WRM

The WRM can be controlled via a telnet interface. For this reason, the Task 9 device shall have a telnet client allowing a user to connect to the WRM for manual configuration.

IP API Interface to WRM

The IP API interface to the WRM consists largely of get, set, send, and transmit commands supplied to calling applications through an API supplied with the WRM. The API utilizes an Ethernet connection to send packets to the WRM issuing commands. The API uses the IP Options field of the IP Header to send the commands.

The API has been implemented and tested utilizing a Microsoft Windows operating system. The supplier developing the Task 9 software will be responsible interfacing with the API on the Task 9 device. The source code for the API will be written utilizing Berkeley Sockets.

In the case where the WRM hardware becomes available but the API has not yet been ported, telnet commands to the WRM may be used to statically configure the WRM which will allow testing of other functionality.

2.3.2.3 Wireless Data Output

In addition to the data content sent from the OBU test and RSU test applications, each data packet shall also contain the following:

- *Sender ID* (either preset OR randomized at start of test)
- *Common Message ID* (to be defined by VSCC members)
- *Message Count* (incrementing short unsigned integer)
- *Broadcast Power*
- *Operating System Time* (nanoseconds)
- *State of Data Logging* (i.e., logging active, logging inactive)
- *Latitude, Longitude, Height* (ellipsoidal)
- *GPS Seconds in Week* (conversion from UTC time to GPS Seconds in Week) (set *GPS Seconds in Week* to zero for devices with no GPS receiver attached)
- *Heading* (from GPS receiver)

2.3.2.4 Data Logging

By default, the Task 9 device shall record all data sent and all data received from surrounding devices via the WRM. In addition, the Task 9 device shall record reception parameters, timing, and statistics associated with each data packet. The recorded reception parameters include:

- *Receiver Signal Strength Indicator (RSSI)*
- *Distance to Sender* (using valid GPS coordinates of sender and receiver)
- *Heading to Sender* (using valid GPS coordinates of sender and receiver)
- *Current GPS Seconds in Week* of receiver
- *Latitude, Longitude, Height* (ellipsoidal) of receiver
- *Heading* (from GPS receiver)
- *Speed* (meters/second, via CAN from host vehicle if receiver)

The Task 9 device shall contain enough storage space to record eight hours of data with five communicating devices in the immediate area transmitting a 100 byte packet once every 100 milliseconds. The 100 byte packet does not include overhead introduced by the storage medium nor does it include the size of the reception parameters.

Members of the VSC consortium plan to modify the OBU test and RSU test applications to support specific application development in the potential future VSC projects. The supplier of the Task 9 software shall write the OBU test and RSU test applications in such a way as to provide a mechanism allowing recording of additional data.

All parameters controllable from the control user interface shall be recorded.

2.3.2.5 Application Display

The Task 9 device shall have a monitor/display and data entry device (such as a keyboard) allowing the user to configure the applications and observe their run-time status.

Control User Interface

The applications shall retain the value of all parameters controllable from the control user interface during a “normal” power cycle.

The control user interface shall allow the user to configure the applications to transmit wirelessly through the WRM upon application execution. This implies the application will also be decoding CAN and/or serial data as appropriate for the application.

The control user interface shall allow the user to adjust all WRM communication parameters described in the WRM interface specification.

The control user interface shall display the IP address of the Task 9 device at the Ethernet port connected to the WRM.

The OBU test and RSU test applications shall each provide a display for configuring the communication parameters. The display shall allow the user to adjust the following communication parameters inherent to the application:

- Periodicity of message transmission
- Size of message being transmitted (append data to standard message using same message ID)
- Destination IP address (as a default, this should be the broadcast IP address)

The OBU test and RSU test applications shall have the ability to initiate, suspend, or terminate a test. The test shall consist of the OBU test or RSU test applications transmitting and recording data as described in previous sections. The length of the test shall be controllable via the control user interface. The user shall be able to control the length of the test by specifying the number of packets sent or the number of seconds of test time. During a test, the Task 9 device shall record data as outlined in sections above.

The OBU test and RSU test applications shall have the ability to specify a filename for recorded test data. The format for the filename shall be <Test Name>_<Test Run>_<GPS Date> where:

- <Test Name> is a field entered by the user
- <Test Run> is an automatically incrementing number incremented after a test has ended (i.e., terminated by the user, maximum number off packet sent, test time exceeded)
- <GPS Date> is derived from the local GPS information.

Real-Time Interface

The OBU test and RSU test applications shall each have an active visual display that shows other Task 9 devices that have communicated with the host Task 9 device. The display shall support at least 10 other Task 9 devices and update the information displayed at least once per second. The text on the display should be easily readable in a vehicular environment. For each device, the following shall be displayed:

- Sender ID
- State of Data Logging
- Most recent Message Count
- Distance to Sender
- Relative Heading to Sender = Heading to Sender (from GPS differencing) – Heading (only to be calculated when host vehicle speed > 5 m/s)
- RSSI
- Speed

3 Application Message Specification

3.1 Introduction

This chapter provides the Message Specification for the Wireless Access in Vehicular Environments (WAVE) application which was developed in accordance with the requirements in Chapter 2.

3.1.1 Scope

This Task 9 Application (T9App) Message Specification describes the format and content of the OBU and RSU message data sent by the T9App to the WAVE Radio Module (WRM) for wireless transmission. The requirements in this document were derived from the requirements in Chapter 2, the Preliminary Vehicle-To-Vehicle Common Message Set (defined in Section 6.5), and the VSCC-defined OBU and RSU message formats defined Sections 6.3 and 6.4.

3.1.2 Application Interfaces

This section describes the T9App interfaces. Figure 3-1 and Figure 3-2 illustrate the OBU and RSU configurations. For the OBU, the T9App inputs data from a GPS receiver and vehicle bus. For the RSU, the T9App inputs data from a GPS receiver (optional) and traffic signal. The T9App extracts selected data, formats the messages defined in Chapter 3.2, and sends the messages to the WRM. The subparagraphs below describe the relevant data transferred over each interface.

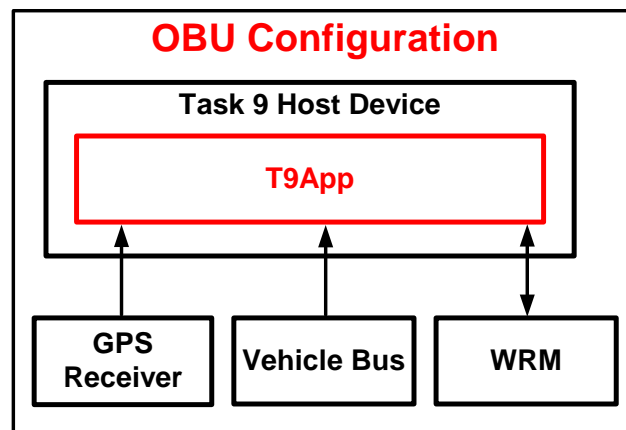


Figure 3-1. OBU Interfaces

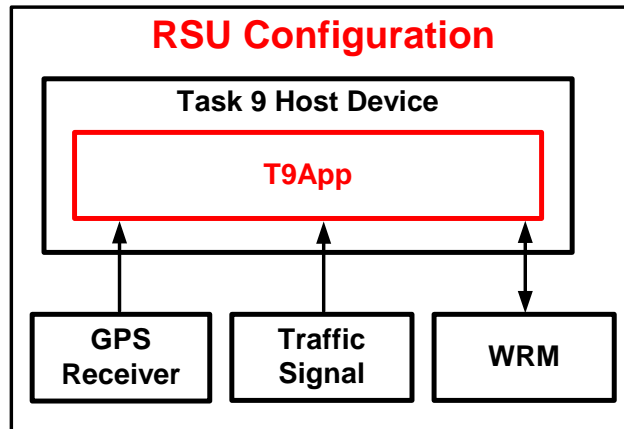


Figure 3-2. RSU Interfaces

3.1.3 GPS Receiver Interface

The T9App receives GPS messages formatted in accordance with the National Maritime Electronics Association (NMEA) 0183 standard [1] over an RS-232 interface. The T9App uses information from the Global Positioning System Fix Data (GPGGA) and Global Positioning Track Made Good and Ground Speed (GPVTG) strings. It obtains date and time from either the UTC Date / Time and Local Time Zone Offset (GPZDA) or the Recommended Minimum Specific GPS/Transit Data string. The T9App operates with GPS receivers that output either string or both.

3.1.4 Vehicle Bus Interface

The T9App receives vehicle data formatted in accordance with Section 6.3 definitions over a Controller Area Network (CAN) bus. The T9App uses information from the Vehicle Velocity, Vehicle Acceleration, and Vehicle Devices messages.

3.1.5 Traffic Signal Interface

The T9App receives a traffic signal message formatted in accordance with the Traffic Signal Interface Specification in Section 6.4 over an RS-232 interface.

3.1.6 WAVE Radio Module Interface

The T9App interfaces to the WRM in accordance with the WRM Interface Specification [2]. The T9App sends the OBU and RSU messages to the WRM as the IP Frame Payload.

3.2 Message Definitions

This section specifies the format of the common message header (which precedes all messages) and the OBU and RSU messages.

3.2.1 Common Message Header

Figure 3-3 illustrates the common message header format and the source of the data in each of the fields. Table 3-1 provides a description of each of the fields.

Bit Byte	8	7	6	5	4	3	2	1(lsb)	SOURCE									
1	Packet Length MSByte								Calculated value									
2	GPS Seconds in Week MSNybble				Packet Length LSNybble (LSBit = 1 Byte)													
3	GPS Seconds in Week ...								Calculated from GPS,GPZDA or GPRMC									
4	GPS Seconds in Week LSByte																	
5	GPS Longitude MSByte								GPS, GPGGA									
6	"																	
7	"																	
8	GPS Longitude LSByte (LSBit = 10**-7 decimal degrees; signed)								GPS, GPGGA									
9	GPS Latitude MSByte																	
10	"																	
11	"								GPS, GPGGA									
12	GPS Latitude LSByte (LSBit = 10**-7 decimal degrees; signed)																	
13	GPS Altitude MSByte								GPS, GPGGA									
14	Altitude (LSBit = 1 cm; unsigned; offset by +1 km)																	
15	Unused				Altitude, LSNybble				GPS, GTVTG (True Heading)									
16	GPS Heading MSByte																	
17	GPS Heading (LSBit = 0.01 degrees; signed; 0 degrees = North)																	
18	Sender ID MSByte								Default value or user input									
19	Sender ID LSByte																	
20	Message Count MSByte								Calculated value									
21	Message Count LSByte																	
22	Unused		Log	Tx Power (LSBit = 1 dBm)					User input (Log, Tx power)									
23	OS Time MSByte																	
24	"								OS High Precision Timer									
25	"																	
26	"																	
27	"																	
28	"																	
29	OS Time LSByte (LSBit = 1 nanosecond)																	

Figure 3-3. Common Message Header Format

Table 3-1. Common Message Header Field Descriptions

Field Name	Description
Packet Length	Total packet length including common message header and message body (i.e., OBU message or RSU message, plus the user-specified number of fill bytes).
GPS Seconds in Week	Number of seconds since the beginning of the week. The range is 0 to 604,799. For units without a GPS receiver, this field is set to 0.
GPS Longitude, Latitude, Altitude	Position information received from the GPS receiver. The altitude is the height above ellipsoid.
GPS Heading	True heading information received from the GPS receiver. This field is set to 0 in RSU messages.
Sender ID	Sender ID of the source of the message. The range is 0 to 65535.
Message Count	Number of messages sent during the current test. The first message is sent with a message count of 1 and the count is incremented in subsequent messages. The count rolls over when it is incremented past 65535.
OS Time	Timestamp from OS clock. The precision and accuracy of the time will vary based on the host device. For the current T9app implementation, the precision will not be accurate to nanoseconds. The timestamp begins at an arbitrary value at the beginning of a test. It rolls over approximately every 28 months.
Log	Logging status. 1 = logging enabled. 0 = logging disabled.
Tx Power	Transmit power setting of the message sender. The range of values is 0 to 20, or 31. A value of 0 to 20 indicates the power setting in dBm. A value of all 1s (i.e., 31) indicates full power.

3.2.2 OBU V2V Safety Message

Figure 3-4 illustrates the SAE V2V Safety Message format that will be used as the OBU message. All signed values are two's complement unless otherwise noted. The figure also lists the source of the data that the T9App will use to populate the message fields. Table 3-2 provides definitions for fields with discrete integer values.

Byte	8	7	6	5	4	3	2	1(lsb)	SOURCE
1	Message Type								Set to 0
2	Temporary ID MSByte								WRM MAC Address
3	Temporary ID...								
4	Temporary ID...								
5	Temporary ID...								
6	Temporary ID...								
7	Temporary ID LSByte								
8	Precision Indicator (Meaning TBD)								Set to 0
9	Longitude of center of vehicle MSByte								GPS, GPVGA
10	"								
11	"								
12	Longitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)								GPS, GPVGA
13	Latitude of center of vehicle MSByte								
14	"								
15	"								GPS, GPVGA
16	Latitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)								
17	Altitude of center of vehicle MSByte								
18	Altitude (LSBit = 1 cm; unsigned; offset by +1 km)								GPS, GPZDA or GPRMC
19	Unused				Altitude LSNybble				
20	UTC Time MSByte								GPS, GPVGT (True Heading)
21	Number of milliseconds since Jan. 1, 2004 at 00:00:00								
22									
23									
24	UTC Time LSByte (LSBit = 0.001 seconds)								CAN, Vehicle Velocity Msg
25	Heading MSByte								
26	Heading (LSBit = 0.01 degrees; signed; 0 degrees = North)								CAN, Vehicle Acceleration Msg
27	Vehicle Speed MSByte								
28	Vehicle Speed LSByte (LSBit = 0.01 m/s; unsigned)								CAN, Vehicle Acceleration Msg
29	Lateral Acceleration MSByte								
30	Longitudinal Acceleration MSNybble				Lateral Acceleration LSNybble (LSBit = 0.01 m/s ² ; signed)				CAN, Vehicle Velocity Msg
31	Longitudinal Acceleration LSByte (LSBit = 0.01 m/s ² ; signed)								
32	Yaw Rate MSByte								CAN, Vehicle Devices Msg
33	Yaw Rate LSByte (LSBit = 0.01 deg/sec; signed)								
34	Throttle Position (LSBit = 0.5% open; unsigned)								CAN, Vehicle Devices Msg
35	Brake Applied Status				Brake Applied Pressure				
36	Steering Wheel Angle MSByte								CAN, Vehicle Devices Msg
37	Steering Wheel Angle LSByte (LSBit = 0.02 degrees; signed)								
38	Headlights		Turn Signal/Hazard Signal		Traction Control State		Anti-Lock Brake State		CAN, Vehicle Devices Msg
39	Unused				System Health				
40	Vehicle Length MSByte								Set to 0
41	Vehicle Width (Upper 2 bits)		Vehicle Length (Lower 6 bits; LSBit = 1 cm; unsigned)						Set to 0
42	Vehicle Width LSByte (LSBit = 1 centimeter)								

Figure 3-4. OBU V2V Safety Message Format

Table 3-2. OBU V2V Safety Message Discrete Field Definitions

Byte	Field Name	Description
35	Brake Applied Status	0000 = All off XXX1 = Left front XX1X = Left rear X1XX = Right front 1XXX = Right rear 1111 = All on
35	Brake Applied Pressure	0000 = Not equipped 0001 = Minimum braking pressure 0010 ... 1111 = Maximum braking pressure
38	Headlights	00 = Off 01 = Daytime running lights 10 = On 11 = Brights
38	Turn Signal/Hazard Signal	00 = Off 01 = Left turn signal 10 = Right turn signal 11 = Hazard signal
38	Traction Control State	00 = Not equipped 01 = Off 10 = On 11 = Engaged
38	Anti-Lock Brake State	00 = Not equipped 01 = Off 10 = On 11 = Engaged
39	System Health	0000 = No faults detected 0001 = Specific error codes ... 1111 = "

3.2.3 RSU Traffic Signal Message

Figure 3-5 and Figure 3-6 illustrate the VSCC defined RSU Traffic Signal Message format that will be used as the RSU message. All signed values are two's complement unless otherwise noted. The figure also lists the source of the data that the T9App will use to populate the message fields. Table 3-3 provides definitions for fields with discrete integer values.

Byte	8	7	6	5	4	3	2	1(lsb)	SOURCE
1	Message Type								Set to 1
2	Temporary ID MSByte								WRM MAC Address
3	Temporary ID...								
4	Temporary ID...								
5	Temporary ID...								
6	Temporary ID...								
7	Temporary ID LSByte								
8	Precision Indicator (Meaning TBD)								Set to 0
9	Longitude of RSU GPS Antenna MSByte								GPS, GPGGA or GUI
10	"								
11	"								
12	Longitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)								GPS, GPGGA or GUI
13	Latitude of RSU GPS Antenna MSByte								
14	"								
15	"								GPS, GPGGA or GUI
16	Latitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)								
17	Altitude of RSU GPS Antenna MSByte								
18	Altitude LSByte (LSBit = 1 cm; unsigned; offset by +1 km)								GPS, GPZDA or GPRMC
19	Unused				Altitude LSNybble				
20	UTC Time MSByte								GPS, GPZDA or GPRMC
21	Number of milliseconds since Jan. 1, 2004 at 00:00:00								
22	UTC Time LSByte (LSBit = 0.001 seconds)								
23	Longitude of Stopping Location #1 MSByte								GUI
24	"								
25	"								
26	Longitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)								GUI
27	Latitude of Stopping Location #1 MSByte								
28	"								
29	"								GUI
30	Latitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)								
31	Altitude of Stopping Location #1 MSByte								
32	Altitude LSByte (LSBit = 1 cm; unsigned; offset by +1 km)								GUI
33	Unused				Altitude LSNybble				
34	Directionality of Stopping Location #1 MSByte								GUI
35	Directionality LSByte (LSBit = 0.01 degrees; signed; 0 degrees = North)								
36	Current State of Traffic Light at Stopping Location #1								Traffic Signal
37	Time Left in Current State of Traffic Light at Stopping Location #1 MSByte								Traffic Signal
38	Time Left LSByte (LSBit = 0.001 seconds) (This field is invalid if current state is red.)								GUI
39	Duration of Yellow State at Stopping Location #1 MSByte								
40	Yellow State Duration LSByte (LSBit = 0.001 seconds)								
41	Longitude of Stopping Location #2 MSByte								GUI
42	"								
43	"								
44	Longitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)								GUI
45	Latitude of Stopping Location #2 MSByte								
46	"								
47	"								
48	Latitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)								

Figure 3-5. RSU Traffic Signal Message Format (Part 1 of 2)

51	Altitude of Stopping Location #2 MSByte	GUI
52	Altitude LSByte (LSBit = 1 cm; unsigned; offset by +1 km)	
53	Unused Altitude LSNybble	
54	Directionality of Stopping Location #2 MSByte	GUI
55	Directionality LSByte (LSBit = 0.01 degrees; signed; 0 degrees = North)	
56	Current State of Traffic Light at Stopping Location #2 (See Below)	Traffic Signal
57	Time Left in Current State of Traffic Light at Stopping Location #2 MSByte	Traffic Signal
58	Time Left LSByte (LSBit = 0.001 seconds) (This field is invalid if current state is red.)	
59	Duration of Yellow State at Stopping Location #2 MSByte	GUI
60	Yellow State Duration LSByte (LSBit = 0.001 seconds)	
61	Longitude of Stopping Location #3 MSByte	GUI
62	"	
63	"	
64	Longitude LSByte (LSBit = 10^{-7} decimal degrees; signed)	GUI
65	Latitude of Stopping Location #3 MSByte	
66	"	
67	"	GUI
68	Latitude LSByte (LSBit = 10^{-7} decimal degrees; signed)	
69	Altitude of Stopping Location #3 MSByte	
70	Altitude LSByte (LSBit = 1 cm; unsigned; offset by +1 km)	GUI
71	Unused Altitude LSNybble	
72	Directionality of Stopping Location #3 MSByte	
73	Directionality LSByte (LSBit = 0.01 degrees; signed; 0 degrees = North)	GUI
74	Current State of Traffic Light at Stopping Location #3 (See Below)	
75	Time Left in Current State of Traffic Light at Stopping Location #3 MSByte	Traffic Signal
76	Time Left LSByte (LSBit = 0.001 seconds) (This field is invalid if current state is red.)	
77	Duration of Yellow State at Stopping Location #3 MSByte	GUI
78	Yellow State Duration LSByte (LSBit = 0.001 seconds)	
79	Longitude of Stopping Location #4 MSByte	GUI
80	"	
81	"	
82	Longitude LSByte (LSBit = 10^{-7} decimal degrees; signed)	GUI
83	Latitude of Stopping Location #4 MSByte	
84	"	
85	"	GUI
86	Latitude LSByte (LSBit = 10^{-7} decimal degrees; signed)	
87	Altitude of Stopping Location #4 MSByte	
88	Altitude LSByte (LSBit = 1 cm; unsigned; offset by +1 km)	GUI
89	Unused Altitude LSNybble	
90	Directionality of Stopping Location #4 MSByte	
91	Directionality LSByte (LSBit = 0.01 degrees; signed; 0 degrees = North)	GUI
92	Current State of Traffic Light at Stopping Location #4 (See Below)	
93	Time Left in Current State of Traffic Light at Stopping Location #4 MSByte	Traffic Signal
94	Time Left LSByte (LSBit = 0.001 seconds) (This field is invalid if current state is red.)	
95	Duration of Yellow State at Stopping Location #4 MSByte	GUI
96	Yellow State Duration LSByte (LSBit = 0.001 seconds)	

Figure 3-6. RSU Traffic Signal Message Format (Part 2 of 2)

Table 3-3. RSU Traffic Signal Message Discrete Field Definitions

Byte	Field Name	Description
36, 56, 74, and 92	Current State of Traffic Light	0 = Green 2 = Yellow 3 = Red

4 Testing System Specification and Architecture

4.1 Introduction

This chapter provides the Software Design Document for the Wireless Access in Vehicular Environments (WAVE) application developed in accordance with the requirements in Chapter 2.

4.1.1 Scope

This section describes the architecture, classes, data structures, and design goals of the Task 9 Application (T9App). It also provides the guidance for implementing future enhancements.

4.2 Architecture

4.2.1 Overview

Figure 4-1 illustrates the T9App architecture. The T9App consists of the WAVETest.exe, the WaveAPI.DLL and the PCAN_USB.DLL. Denso developed the software for the WAVETest.exe and WaveAPI.DLL. Grid Connect supplied the PCAN_USB.DLL with their USB Controller Area Network (CAN) adapter.

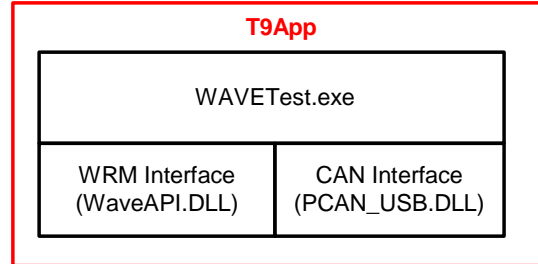


Figure 4-1. T9App Architecture

4.2.2 External Interfaces

Figure 4-2 illustrates the T9App external interfaces. The T9App interfaces to a GPS receiver, a vehicle bus (OBU only), a traffic signal (RSU only), and the Wave Radio Module (WRM). The subparagraphs below describe the software supporting each of these interfaces.

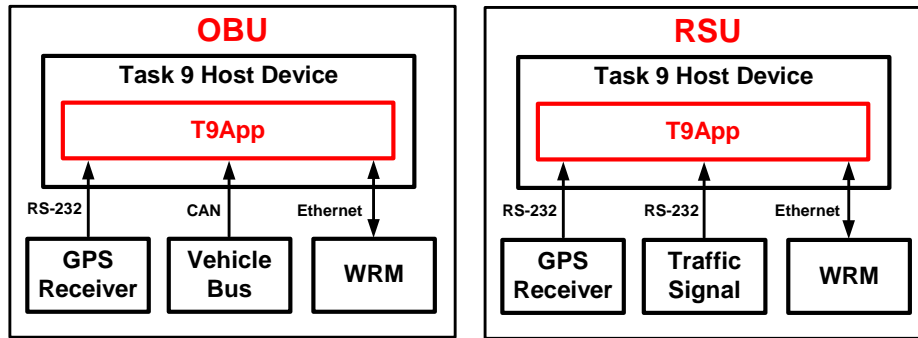


Figure 4-2. T9App External Interfaces

4.2.2.1 GPS/Traffic Signal Interface

The T9App interfaces to the GPS and Traffic Signal using the standard Microsoft Windows serial port drivers.

4.2.2.2 Vehicle Bus Interface

The T9App interfaces to the vehicle bus using the PCAN_USB.DLL supplied by Grid Connect. The Task 9 Host Device (HD) must also have the Grid Connect PCAN_USB.SYS driver installed.

4.2.2.3 WRM Interface

The T9App interfaces to the WRM using the WaveAPI.DLL originally developed for VSCC under Task 6D.

4.2.3 User Interface

4.2.3.1 Screen Shots

The T9App User Interface is a collection of configuration dialogs and a real time Test Display dialog screen. See Figure 4-3 for screen shots of the configuration dialogs and Figure 4-4 for the Test Display dialog.

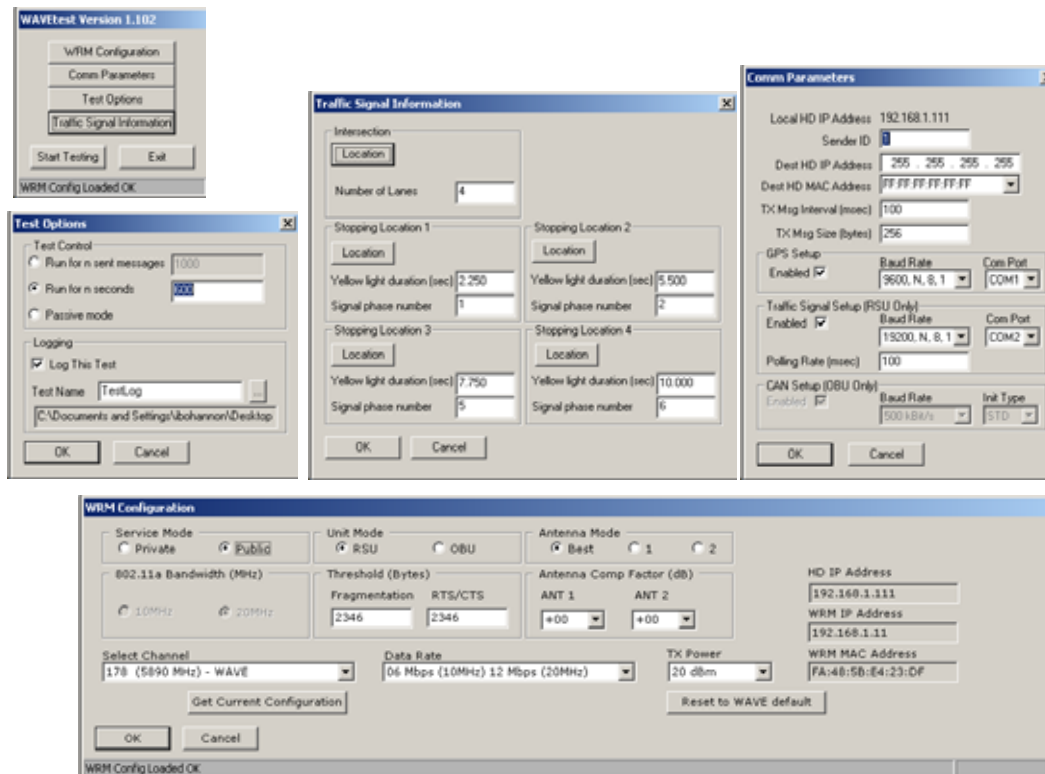


Figure 4-3. Configuration Dialogs Screen Shots

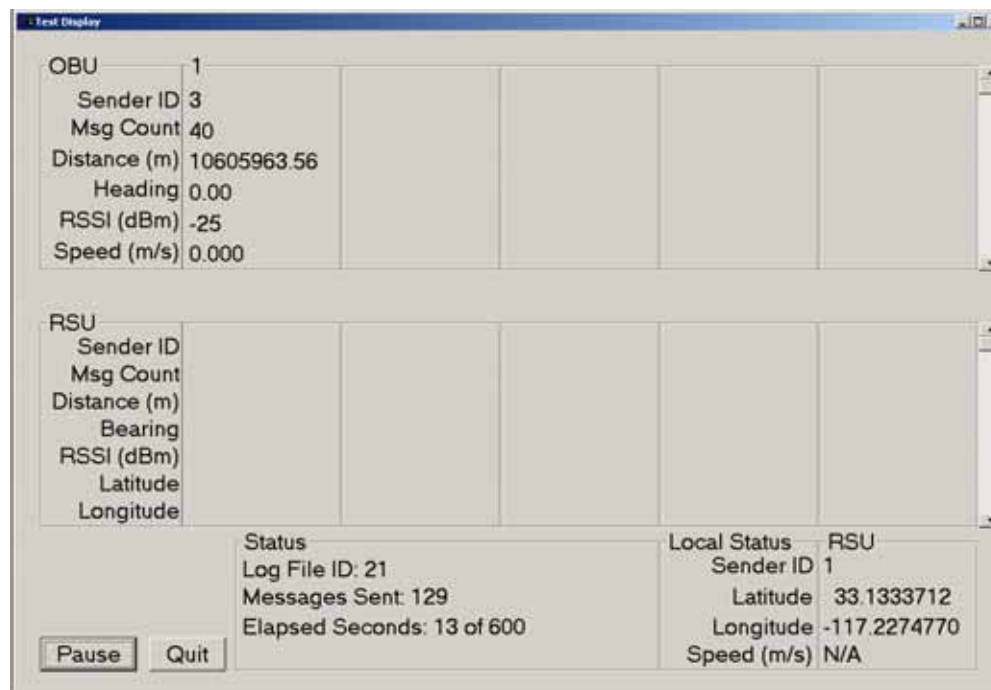


Figure 4-4. Test Display Dialog Screen Shot

4.2.3.2 T9App Dialog to Class Relationships

Table 4-1 summarizes the classes for the main application and the dialogs. For all dialogs, a Microsoft Foundation Class (MFC) based GUI class controls the display, data entry, and range checking. For configuration dialogs, a second information class (generic class type) stores the data in a portable manner.

Table 4-1. T9App Dialog Class Summary

Dialog/App	GUI Class	Information Class
Main Windows App	CWaveTestApp	N/A
Main Dialog	CWaveTestDlg	N/A
WRM Configuration Dialog	CVSCCGUIDlg	CWrmControl
Comm Parameters Dialog	CCommParamDialog	CCommParams
Test Options Dialog	CTestOptionsGui	CTestOptions
Traffic Signal Information Dialog	CRsuParamtersDialog	CSignalInfo
Location Information Dialog	CLocationDialog	CLocationInfo
Test Display Dialog	CTestStatusDialog	N/A

4.2.4 Test Control

4.2.4.1 Class Architecture

The CTestStatusDialog class controls test execution. Figure 4-5 illustrates the high level architecture in a Unified Modeling Language (UML) format (see www.uml.org for more information). The CTestStatusDialog class owns control classes for each hardware interface (i.e., CGpsControl, CSignalControl, CWrmControl, and CCanControl). The CSerialPort class manages the serial ports for the GPS and Traffic Signal interfaces.

The ThisDevice object maintains local device information and the CDeviceDB class maintains OBU or RSU message data received from each unique sender ID. The CTestLog class writes the test log.

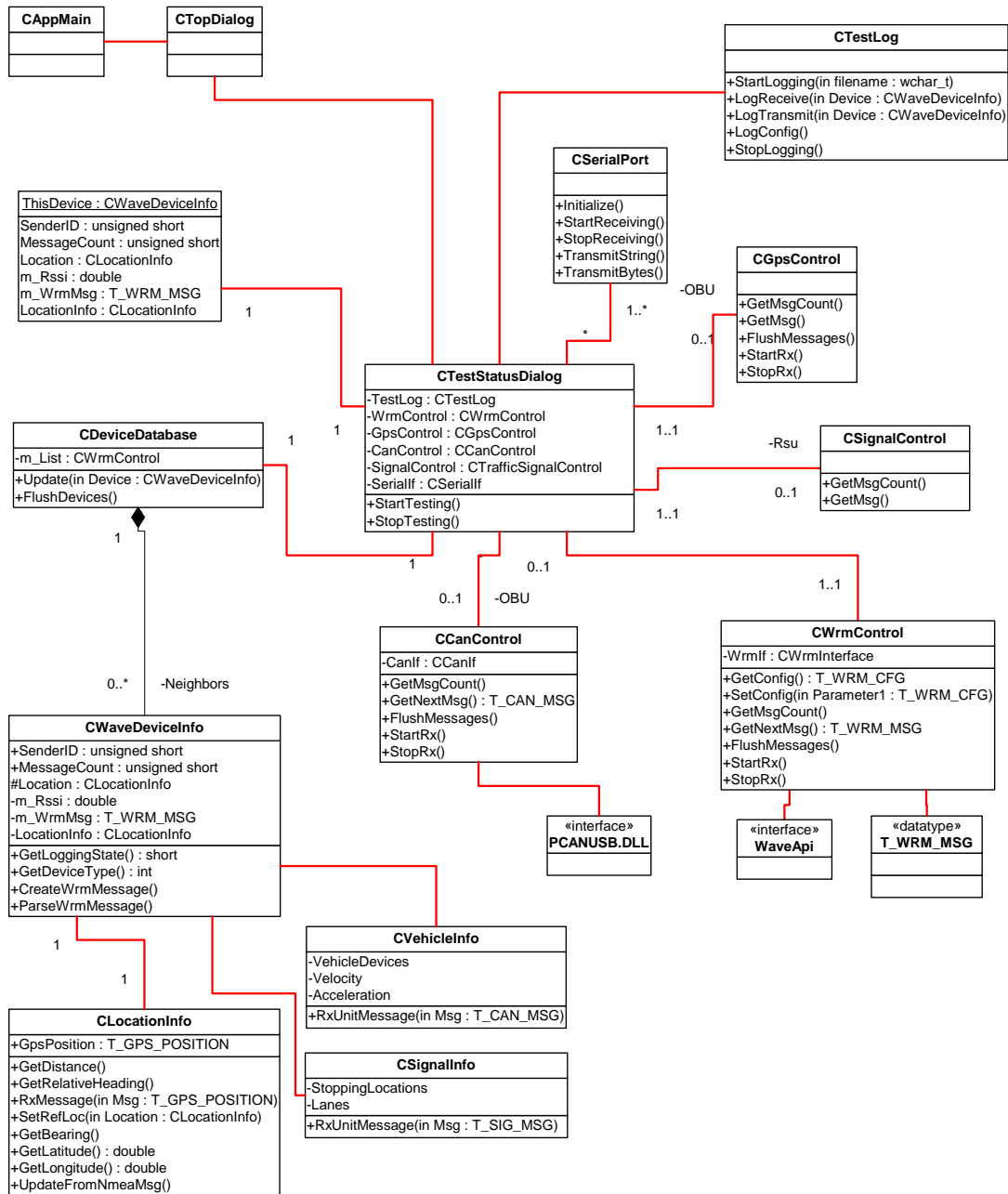


Figure 4-5. T9App Test Execution Architecture (UML Diagram)

4.2.4.2 Test Processing

The CTestStatusDialog object maintains three timers, a Tx Message Interval Timer, a Screen Refresh Timer, and an optional Traffic Signal Service Timer. The T9App sets the Tx Message Interval Timer to the user-entered value on the Comm Parameters screen. The Screen Refresh Timer is always set to one second. The T9App sets the Traffic Signal Service Timer to the user-entered value on the Comm Parameters screen.

Upon expiration of the Tx Message Interval Timer, the CTestStatusDialog object processes all of the messages received from the GPS and Vehicle Bus since the last timer expiration. It formats the OBU or RSU message from the latest information and sends the message to the WRM for transmission and to the CTestLog object for logging. It also processes all messages received from the WRM. It sends the data along with the required reception parameters (e.g., RSSI, timestamp) to the CDeviceDatabase object for entry in the database and to the CTestLog object for logging.

Upon expiration of the Screen Refresh Timer, the T9App queries the CDeviceDatabase and updates the Test Display dialog with the latest information received from remote devices, in order of first unit to the last. The T9App also updates the local device information and current testing statistics.

Upon expiration of the Traffic Signal Service Timer, the T9App sends out its periodic refresh message and updates its traffic signal information with all messages received since the last timer expiration.

The timer processing continues until the user-defined test completion criteria is met, or until the user presses the quit button.

4.3 Class Descriptions

Table 4-2. T9App Class Summary lists the classes used for test execution along with their type and purpose. The class types are control, container, and processing. Some classes are both container and processing. Control classes abstract an underlying API and provide a C++ interface for the T9App architecture. Container classes provide wrapper functions for getting and setting various T9App data stores and messages. Processing classes use a combination of control and/or container classes to implement application functionality and features.

Table 4-2. T9App Class Summary

Class	Class Type	Purpose
CCanControl	Control	Uses PCANUSB.DLL to read messages from CAN bus, queues CAN messages for processing.
CCanMsg	Container	Wraps CAN messages received from CAN I/F.
CCommParams	Container	Holds communication related test parameters.
CDeviceDatabase	Container	Holds list of remote devices from which messages were received during a test run.
CGpsControl	Control	Reads and queues GPS NMEA messages.
CGpsPosition	Container	Wraps and parses NMEA message data.
CLocationInfo	Container/Processing	Performs navigation calculations and data store for location related information.

Class	Class Type	Purpose
CSerialPort	Processing	Provides a C++ wrapper for the Microsoft Standard Serial Port Library, and provides a thread based event handler w/ dispatch. The GPS and Traffic Signal Control modules use this class.
CSignalControl	Control	Reads / polls and queues traffic signal messages.
CSignalInfo	Container	Holds traffic signal stopping location and current state information.
CTestLog	Processing	Logs all transmitted and received data.
CTestOptions	Container	Holds test options related parameters.
CTestStatusDialog	Main Control / Processing	Runs test case and displays test status.
CVehicleInfo	Container	Holds all vehicle related information.
CWaveDeviceInfo	Container	Provides master container for all WAVE device information, OBU and RSU. Also holds generic WAVE device information, not specific to RSU or OBU.
CWrmControl	Control	Reads/queues and transmits WRM messages.
CWrmMsg	Container/Processing	Provides generic wrapper for transmitting WRM messages, also provides parsing and encoding functions for over the air formats.

4.3.1 Class Descriptions

The following subparagraphs summarize each T9App class, and describe each class's public methods and attributes.

4.3.1.1 Class CCanControl

The CCanControl class encapsulates the CAN USB interface. It starts a receive processing thread, and buffers CAN messages in real time for deferred processing. See Table 4-3 for details.

Table 4-3. CCanControl Methods and Attributes

Method/Attribute	Description
unsigned int GetCanBusPollingInterval();	Returns value of the CanBusPollingInterval.
CCanMsg& GetNextRx();	Retrieves the head of the message list. If the list is empty, the results are undefined.
int GetRxCount();	Returns message count of CAN message list.
int Init(int CanBaud, int CanFrame);	Attempts to initialize CAN Interface via the PCAN_USB DLL. Returns error codes if unable to load or initialize DLL.
BOOL InitRxThread();	Creates and starts execution of thread to check for and receive CAN messages from PCAN_USB.DLL. Places received messages into the message list.
SetCanBusPollingInterval(unsigned int interval);	Sets the interval to check the CAN bus status.
StopRxThread();	Stops the receive thread; preserves all unprocessed CAN messages in the queue.
BOOL m_Initialized;	Tracks current init state of underlying CAN Interface DLL.

4.3.1.2 Class CCanMsg

The CCanMsg class encapsulates a Rx CAN message. It parses all supported CAN messages (currently 3), and fills out a preconfigured CVehicleInfo object when the Update () method is called. See Table 4-4 for details.

Table 4-4. CCanMsg Methods and Attributes

Method/Attribute	Description
TPCANMsg* GetRxPacket()	Gets next CAN message.
SetTheVehicle(CVehicleInfo* Vehicle)	Sets the vehicle object that will be accessed when the UpdateVehicle method is called to parse the CAN Messages.
UpdateVehicle	Parses the privately held CAN message and calls the appropriate set methods in the preconfigured Vehicle object for each data item read from the CAN message.

4.3.1.3 Class CCommParams

The CCommParams class owns the communication parameters for the T9App. The config dialogs access the members of this class pre-test, and the test status dialog uses this class to set up the test run. See Table 4-5 for details.

Table 4-5. CCommParams Methods and Attributes

Method/Attribute	Description
BYTE* GetDestMacAddress(); SetDestMacAddress(BYTE a1,BYTE a2, BYTE a3, BYTE a4, BYTE a5,BYTE a6);	Gets or sets Destination MAC address.
friend ostream &operator<<(ostream&, const CCommParams&); friend istream &operator>>(istream&, CCommParams&); PrintHeaders(ostream &output); bool ConfigLoaded();	These methods support saving, loading, and logging of the configuration.
USHORT GetTrafficSignalPollRate(); SetTrafficSignalPollRate(USHORT rate);	Gets traffic signal polling rate.
SetHostIpAddress(BYTE a1, BYTE a2, BYTE a3, BYTE a4); GetHostIpAddress(BYTE *a1, BYTE *a2, BYTE *a3, BYTE *a4); CString& GetHostIpAddressString();	Sets and gets the Host IP Address parameters – the T9App automatically gets the IP address from the OS.
SetDestIpAddress(unsigned long); GetDestIpAddress(BYTE*a1, BYTE *a2, BYTE *a3, BYTE *a4); GetDestIpAddress(BYTE* address); unsigned long GetDestIpAddress(); CString& GetDestIpAddressString();	Sets and gets the Destination IP address.
unsigned short GetSenderId(); int SetSenderId(CString& val); SetSenderId(USHORT val); CString& GetSenderIdString();	Gets and sets the Sender ID for the over the air message header.
int GetMessageRateMsec(); int SetMessageRateMsec(CString& val); SetMessageRateMsec(int val); CString& GetMessageRateMsecString();	Gets and sets the rate at which messages are sent over the air.
int SetMessageSize(CString& val); SetMessageSize(int val); CString& GetMessageSizeString();	Sets the minimum over the air message size. The T9App automatically increases this value to the actual message size if it is too low.
ResetDefaults();	Resets all values to defaults.

Method/Attribute	Description
unsigned int m_CanBaudIndex; unsigned int m_CanInitType; BOOL m_CanEnabled;	CAN I/F initialization parameters.
unsigned int m_GpsBaudIndex; unsigned int m_GpsPort; BOOL m_GpsEnabled;	GPS Serial Port initialization parameters.
unsigned int m_TrafficSignalBaudIndex; unsigned int m_TrafficSignalPort; BOOL m_TrafficSignalEnabled; USHORT m_TrafficSignalPollRate;	Traffic Signal initialization parameters.

4.3.1.4 Class CDeviceDatabase

The CDeviceDatabase class maintains an up-to-date list of all remote WAVE devices from which messages were received. This class contains 2 CLists, 1 for OBUs and 1 for RSUs. See Table 4-6 for details.

Table 4-6. CDeviceDatabase Methods and Attributes

Method/Attribute	Description
ResetDatabase()	Iterates thru OBU and RSU lists and removes each item.
CList<CWaveDeviceInfo, CWaveDeviceInfo&> m_RsuList; CList<CWaveDeviceInfo, CWaveDeviceInfo&> m_ObuList;	CList Template, holds a list of references to OBU/RSU class WAVE devices
void UpdateRemoteDevice(CWaveDeviceInfo& NewDevice);	Updates the remote device list by adding the device or updating the data. This function is called each time an over the air message is received and successfully parsed into a new CWaveDeviceInfo object.

4.3.1.5 Class CGpsControl

The CGPSControl class processes National Maritime Electronics Association (NMEA) format messages from a GPS device plugged into a local configurable serial port. The CTestStatusDialog (CDialog) object owns the serial port class, due to serial class implementation. The CTestStatusDialog forwards all messages from the serial port to this class for actual processing. See Table 4-7 for details.

Table 4-7. CGpsControl Methods and Attributes

Method/Attribute	Description
BOOL Initialize(CWnd* pPortOwner, int port, int baudindex);	Called by the object owner to attempt to initialize the serial port, returns FALSE if initialization fails.
OnCommunication, OnErrDetected, OnRxFlagDetected	Called by TestStatusDialog when rx char, error, or rxflag event is received.
void ProcessMessages();	Processes the first message the GPS Control class may have in its list.
Void SetLocalDevice(CWaveDeviceInfo *Device);	Sets the device to be updated when the NMEA messages are parsed. .
StartRxThread();	Starts the serial ports receive thread, if the serial port initialized without error.
StopRxThread();	Stops the serial ports receive thread, used when testing is paused or terminated.
CSerialPort* m_Serial;	Pointer to serial object controlled by this class.

4.3.1.6 Class CGpsPosition

The CGpsPosition class represents a decoded GPS position, and provides methods to decode NMEA messages into a GPS position. Each local WAVE device's ClocationInfo will have a corresponding CGpsPosition object. See Table 4-8 for details.

Table 4-8. CGpsPosition Methods and Attributes

Method/Attribute	Description
UpdateFromNmeaMsg(CString &data);	Given the NMEA message in a string, this method fills out the details of the GPS Position and sets the appropriate m_Received_XXX flags.
double m_Heading;	GPS Heading.
BOOL m_Received_GGA;	Holds flag that is set to true if this object processed a NMEA GPGGA message. Note more than one received flag may be set during life of this object.
BOOL m_Received_RMC;	Holds flag that is set to true if this object processed a NMEA GPRMC message. Note more than one received flag may be set during life of this object.
BOOL m_Received_VTG;	Holds flag that is sets to true if this object processed a NMEA GPVTG message. Note more than one received flag may be set during life of this object.

Method/Attribute	Description
BOOL m_Received_ZDA;	Set to true if this object processed a NMEA GPZDA message. Note more than one received flag may be set during life of this object.
double m_utc;	Parsed UTC Time in fixed point format: Hhmmss.ss
double m_latitude;	Parsed Latitude of position, ddmm.mmm
char m_latNS;	Parsed Latitude North/South qualifier. Either 'N' or 'S'
double m_longitude;	Parsed Longitude of position, ddmm.mmm
char m_longEW;	Parsed Longitude east/west qualifier. Either 'E' or 'W'
int m_fixQual;	Parsed fix quality.
int m_sats;	Parsed number of satellites in view
double m_hdop;	Parsed relative accuracy of horizontal position.
double m_altitude;	Parsed units above mean sea level.
char m_altUnits;	Parsed units qualifier for altitude. 'M' for meters.
double m_hogae;	Parsed height of geoid above WGS84 ellipsoid (HOGAE) in meters.
char m_hogUnits;	Parsed units qualifier for HOGAE. 'M' for meters, this is the units for the m_hogae
char m_csum[1];	Parsed NMEA checksum used by program to check for GPS transmission errors. Not currently implemented.
CString m_utctime;	UTC Time in String format.
int m_month, m_day, m_year;	Parsed UTC month, day, and year.
int m_hours, m_minutes, m_seconds, m_hundreths;	Parsed UTC hours, minutes, seconds, and hundredths.

4.3.1.7 Class CLocationInfo

The CLocationInfo class owns a GPS position and provides navigational calculations and formatting retrieval methods for working with location information. See Table 4-9 for details.

Table 4-9. CLocationInfo Methods and Attributes

Method/Attribute	Description
unsigned char GetAltitudeLsb();	Gets LSB part of Altitude for the over the air message.
unsigned short GetAltitudeMsb();	Gets MSB part of Altitude for the over the air message.
CString& GetAltitudeStr();	Gets Altitude.
Double GetBearing(CLocationInfo& Destination);	Computes the initial bearing from the local location to the given destination.

Method/Attribute	Description
CString& GetBearingString(CLocationInfo& Destination);	Returns the initial bearing in a formatted string, suitable for GUI display or logging.
CString& GetFormattedLatStr(); double GetLatitude();	Latitude access methods. Floating point, string.
CString& GetFormattedLongStr(); double GetLongitude();	Longitude access methods, floating point, string. Etc.
DWORD GetGpsSecondsInWeek();	Returns pre-computed GPS seconds in week. Returns zero if GPS messages containing this information haven't been received yet.
signed short GetHeadingInt();	Returns Heading in signed integer format, suitable for over the air transmission.
CString& GetHeadingStr(); unsigned short GetHeadingUINT();	Heading access methods of various combinations.
unsigned char GetPrecision();	Returns the precision of the OS timestamp.
double GetRelativeDistance(location) CString& GetRelDistanceStr(location)	Computes distance from object to given location.
double GetRelativeHeading(CLocationInfo& Location) CString& GetRelHeadingStr(CLocationInfo& Location)	Computes and returns relative heading from object to given location in different formats.
GetUTCTimeSinceJan12004(PTIME_MSEC Value); unsigned char GetUTCTimeSinceJan12004Lsb(); unsigned long GetUTCTimeSinceJan12004Msb();	Returns the number of milliseconds that have elapsed since UTC Time January 1 st , 2004. This method uses the current UTC time retrieved from GPS receiver as the starting point. If UTC time has not been received from the GPS yet, or if the GPS is not enabled, this function returns 0 values in the TIME_MSEC structure.
PrintHeaders(ostream &output);	Outputs to stream the field labels for each element that the overloaded input and output functions read/write.
PrintHeaders(ostream &output, CString prefix);	Outputs to stream the field labels with an optional prefix for each element that the overloaded input and output functions read/write.
ResetLocationDefaults();	Resets all run-time values to zero.
SetAltitude(unsigned short msb, unsigned char lsb);	Sets Altitude.
void SetAltitude(double alt);	Sets Altitude.
int SetAltitudeStr(CString &val);	Sets Altitude.

Method/Attribute	Description
SetGpsSecondsInWeek(DWORD Seconds);	Sets GPS seconds in week..
SetHeading(double Heading); SetHeading(signed short Heading); int SetHeadingStr(CString& heading);	Sets Heading.
int SetLatitude(CString &val); SetLatitude(double val);	Sets Latitude.
int SetLongitude(CString &val); SetLongitude(double val);	Sets Longitude.
SetPrecision(unsigned char Precision);	Sets the precision of the OS timestamp.
SetUtcTimeSinceJan12004(TIME_MSEC time);	Sets UTC time from the over the message.
UpdateLocationFromGpsPosition(CString& NmeaMsg);	Updates the location from information in the NMEA message.
friend ostream &operator<<(ostream&, const CLocationInfo&);	Writes the current values of the member data to the given ostream, used for logging configuration and saving configuration.
friend istream &operator>>(istream&, CLocationInfo&);	Reads member data from the given input stream. Used to read application configuration from disk.

4.3.1.8 Class CSerialPort

The CSerialPort class encapsulates the Microsoft serial port interface with a C++ methods and a threaded state machine for handling simultaneous transmit and receive. The GPS control class and the traffic signal control class use this class. The T9App configures the serial interfaces to send all events to the CTestStatusDialog window. When the CTestStatusDialog receives events, it forwards them to the appropriate control class depending on how the user configured the serial ports. For example, if the user configured COM1 for GPS, the CTestStatusDialog forwards all COM1 port event messages to the GPS control class. See Table 4-10 for details.

Table 4-10. CSerialPort Class Methods and Attributes

Method/Attribute	Description
GetCommEvents();	Returns the configured communications event masks.
GetDCB();	Returns the device control block created at initialization.
GetWriteBufferSize();	Returns write buffer size that was specified in InitPort() call.
InitPort(PortOwner, portnr, baud, parity, databits,, stopsbits = 1, dwCommEvents, nBufferSize)	Opens the specified serial port with the specified baud rate, parity, stop bits, and tx buffer sizes.
KillThread();	Kills the comm. Port receive thread.
RestartMonitoring();	Stops and re-starts comm. port receive thread.
StartMonitoring();	Starts serial port receive thread.
StopMonitoring();	Pauses the serial port receive thread, doesn't kill it.
WriteToPort(char* string);	Adds the specified character string to the tx queue and begins transmission.
WriteToPort(unsigned char* string, unsigned int len);	Adds the specified byte buffer to the tx queue begins transmission.

4.3.1.9 Class CSignalControl

Methods and Attributes

The CSignalControl class controls a traffic signal plugged into the serial port this object has been given a reference to. See Table 4-11 for details.

Table 4-11. CSignalControl Methods and Attributes

Method/Attribute	Description
USHORT GetPollingRate();	Returns current traffic signal polling rate in milliseconds.
BOOL Initialize(CWnd *pPortOwner, int port, int baudindex);	Initializes serial port according to the given parameters. Returns false if init failed.
SetPollingRate(USHORT rate);	Sets traffic signal polling interval in milliseconds.
StartRxThread();	Starts receive thread if it isn't already running.

Method/Attribute	Description
void StateMachine(enum TS_EVENT Event, void *data);	Traffic Signal state machine entry point. When the traffic signal interface is enabled, the state machine must be clocked by calling this function a tick event parameter and tick interval value as a data. When serial data is received, this function must be called with TSE_RX_DATA event, and a character in data portion.
StopRxThread();	Stops the receive thread if it is running.
TransmitPeriodicRequest();	Transmits poll message to traffic signal.
CSerialPort* m_Serial;	Serial port this object controls.

State Machine

The CSignalControl implements the Traffic Signal Interface with a simple state machine. The state machine implements the initialization sequence in accordance with the timing requirements. Table 4-12 lists the events processed by state machine and describes each event.

Table 4-12. CSignalControl State Machine Events

Event	Description
TSE_NONE	No event.
TSE_INIT_OK	Initialization sequence succeeded.
TSE_RX_DATA	Serial port has received a character.
TSE_TX_DATA	Serial port TX has completed.
TSE_RX_FLAG	Serial port RX flag indicated.
TSE_ERROR	Serial port error indicated.
TSE_CTS	Serial port clear to send indication.
TSE_TICK	Owner is indicating that time has elapsed.
TSE_PROCESS	Indicates queued messages should be processed.
TSE_SHUTDOWN	Owner is shutting down.

4.3.1.10 Class CSignalInfo

The CSignalInfo class contains the traffic signal information. The CWaveDeviceInfo owns this object which contains valid data for a RSU. See Table 4-13 for details.

Table 4-13. CSignalInfo Methods and Attributes

Method/Attribute	Description
unsigned short GetDurationOfYellowLightMsec(int signal); unsigned short GetSignalPhaseNumber(int signal); unsigned char GetSignalState(int signal); unsigned short GetSignalStateTimeLeftMsec(int signal);	Obtain signal information from over the air messages.
CString& GetLanesString();	Gets the number of lanes in a string format.
friend ostream &operator<<(ostream&, const CSignalInfo&); friend istream &operator>>(istream&, CSignalInfo&); PrintHeaders(ostream &output);	Writes T9App configuration to disk or to test log.
ResetSignalDefaults();	Sets all signals to red state and time left to 0.
int SetLanes(CString &lanes);	Set the number of lanes.
SetSignalState(int Phase, SIGNAL_STATE State, int SecondsLeft);	Set current signal state of a remote RSU.
unsigned char GetSignalState(int signal); unsigned short GetSignalStateTimeLeftMsec(int signal); unsigned short GetSignalPhaseNumber(int signal); unsigned short GetDurationOfYellowLightMsec(int signal);	Obtain signal information from over the air messages.
unsigned short m_SignalPhaseNumber[MAX_SIGNALS];	Configured phase number for each signal.
SIGNAL_STATE m_SignalState[MAX_SIGNALS];	Traffic light state for each signal.
unsigned short m_SignalStateTimeRemainingMsec[MAX_SIGNALS];	Configured time remaining for each signal.
CLocationInfo m_StoppingLocation[MAX_SIGNALS];	Coordinates of stopping locations.
unsigned short m_YellowLightDurationMsec[MAX_SIGNALS];	Yellow light duration for each signal

4.3.1.11 Class CTestLog

The CTestLog class writes entries to the log. It is implemented with the ANSI streams library. See Table 4-14 for details.

Table 4-14. CTestLog Methods and Attributes

Method/Attribute	Description
void LogReceive(CWrmMsg& msg, CWaveDeviceInfo& receiver);	Logs a receive entry. The function parses the message into a CWaveDeviceInfo object, queries the object values, and logs them to the current stream. The receiver information is a required parameter to enable logging of reception information.
LogTestParameters(CcommParams, CTestOptions, CwaveDeviceInfo, CwrmControl)	Writes T9App configurable parameters to current test log file.
void LogTransmit(CWrmMsg& msg);	Logs a transmit entry. The function reparses the message back into a CWaveDeviceInfo object, queries the sent values, and logs them to the current stream.
int StartLogging(CString TestName, CString Path, CString GpsDate, int pass);	Creates a candidate filename from a test name, file path, and pass number. Tries to open that filename read-only. If that filename exists, the function increments pass number and tries again until the test log filename is unique. Next, the test log filename is opened for writing and the final pass number is returned to calling function.
StopLogging();	Closes log file, forcing it to be written to disk.

4.3.1.12 Class CTestStatusDialog

Methods

Section 4.2.4 describes the CTestStatusDialog class. See Table 4-15 for the method description.

Table 4-15. CTestStatusDialog Methods and Attributes

Method/Attribute	Description
SetCommParams(CCommParams &Params);	Sets pointer to communication parameters object owned by Test App.
SetLocalDevice(CwaveDeviceInfo)	Sets local CWaveDeviceInfo information to be used for each test pass.
SetTestOptions(CTestOptions &TestOptions);	Sets pointer to test options configuration object owned by Test App.
SetWrmControl(CWrmControl *WrmControl);	Sets pointer to WRM control object owned by Test App.

Processes

Table 4-16 lists all of the threads and timers that the CTestStatusDialog class owns. The CTestStatusDialog uses timers to meet the processing requirements. The threads enable the test display to update without slowing down data events for other functions. The CTestStatusDialog starts and stops the threads and timers for each test pass.

Table 4-16. CTestStatusDialog Processes

Process Name	Purpose	Type
CcanControl.Rx_thread	Checks for the presence of a CAN message. Queues any messages into CcanControl's message list.	Polling Thread. 10 millisecond sleep between polling if CAN bus is idle. Not configurable.
CwrmControl::rx_thread	Checks for and retrieves WRM messages from WaveApi.dll.	Polling Thread. 10msec sleep between polling if WRM is idle. Not configurable.
TestStatusDialog::GpsSerial.rx_thread	Serial event handler/dispatcher.	Event Driven Thread
TestStatusDialog::ProcessMessages	Sends periodic updates. Processes all WRM RX messages, CAN messages, and GPS Messages.	Timer executes at user defined interval.
TestStatusDialog::SignalSerial.rx_thread	Serial event handler/dispatcher.	Event Driven Thread
TestStatusDialog::StartTesting	Auto starts the first test pass when user clicks Start testing from main dialog.	10-millisecond timer single shot timer, not configurable.
TestStatusDialog::TrafficSignalPolling	Clocks the traffic signal interface state machine.	Timer executes at user defined polling rate
TestStatusDialog::UpdateDisplay	Updates GUI at a regular interval, prevents screen flashing & excessive processor utilization for GUI updates.	1-second timer, not configurable.

4.3.1.13 Class CVehicleInfo

The CVehicleInfo class contains all vehicle information. It obtains local information from the CAN bus and remote information by parsing an OBU Vehicle to Vehicle (V2V) message received over the air. Since the requirements for logging vehicle information and transmitting it over the air are slightly different, 2 methods are provided for each vehicle attribute; one for displaying it or logging to disk, and another to format values for over the air transmission. The CVehicleInfo class also provides two methods for setting values, one to set values from a received OBU message and another to set values from a CAN message (in different formats). See Table 4-17 for details.

Table 4-17. CVehicleInfo Methods and Attributes

Method/Attribute	Description
unsigned char GetBrakeInfo();	Gets the brake-applied status in upper nibble, and brake-applied pressure in lower nibble.
double GetLatAccel(); signed short GetLatAccelShort();	Gets Lateral Acceleration
Double GetLongAccel(), signed short GetLongAccelShort()	Gets Longitudinal Acceleration.
Unsigned char GetSignalAndControlStatus();	Gets: Headlight status in bits 7,6 Turn signal/hazard in bits 5,4 Traction control in bits 3,2 Anti-lock brake state in bits 1,0
double GetSteeringWheelAngle(); Signed short GetSteeringWheelAngleShort();	Gets current steering wheel angle as a signed number from -180.00 to +180.00.
unsigned char GetSystemHealth();	Gets system health bits in the lower nibble.
double GetThrottlePos();	Gets throttle position in same precision as CAN bus message format (0.5%).
unsigned short GetVehicleLength();	Gets 14 bits of vehicle length in cm.
unsigned short GetVehicleWidth();	Gets 10 bits of vehicle width in cm.
Double GetVelocity(), Unsigned int GetVelocityUINT(), CString& GetSpeedStr();	Gets current vehicle speed in meters per seconds.
double GetYawRate(); signed short GetYawRateShort();	Gets vehicle yaw rate in degrees as a signed number. LSB = 0.01 deg/sec.
ResetVehicleDefaults();	Resets all values to starting state valid.
SetBrakeInfo(BYTE BrakeInfo);	Sets the brake-applied status (in upper nibble), and the brake-applied pressure (lower nibble).
SetLatAccel(BYTE msb, BYTE lsb); SetLatAccelImmSec(BYTE msb, BYTE lsb);	"
SetLongAccel(BYTE msb, BYTE lsb); SetLongAccelImmSec(BYTE msb, BYTE lsb);	"

Method/Attribute	Description
SetSignalAndControlStatus(BYTE SignalAndControlStatus);	Sets parameters in OBU VTV message and CAN format: Headlight status in bits 7,6 Turn signal/hazard in bits 5,4 Traction control in bits 3,2 Anti-lock brake state in bits 1,0
SetSteeringWheelAngle(signed short Angle); SetSteeringWheelAngle(double Angle);	"
SetSystemHealth(BYTE SystemHealth);	Sets value from lower nibble.
SetThrottlePos(BYTE ThrottlePos);	Sets throttle position from CAN Message or OBU VTV message
SetVehicleLength(BYTE VehicleLengthMsb, BYTE VehicleLengthLsb);	Currently initialized to zero. Not settable by user or CAN message. Only the over the air message parser calls this function.
SetVehicleWidth(BYTE VehicleWidthMsb, BYTE VehicleWidthLsb);	Currently initialized to zero. Not settable by user or CAN message. Only the over the air message parser calls this function.
SetVelocity(WORD Speed); SetVelocity(BYTE msb, BYTE lsb);	2 functions to support setting this value from either a received CAN message or and OBU VTV message.
SetYawRate(signed short YawRate); SetYawRate(BYTE msb, BYTE lsb);	"
friend ostream &operator<<(ostream&, const CVehicleInfo&);	Logs and saves T9App configuration.
friend istream &operator>>(istream&, CVehicleInfo&);	"

4.3.1.14 Class CWaveDeviceInfo

The CWaveDeviceInfo class contains all information about a WAVE device, either OBU or RSU. This class provides methods to get/set all attributes including some formatting methods suitable for displaying WAVE device info on dialogs. This class has overloaded streams operators so that device information can be read or written from any streams type. This class also contains a static instance of a vehicle and a traffic signal object. This class maintains the current RSSI, Message Counts, Sender ID, and other overhead information required for each WAVE device. Note T9App uses this object for storing information about both local and remote WAVE devices. See Table 4-18 for details.

Table 4-18. CWaveDeviceInfo Methods and Attributes

Method/Attribute	Description
bool ConfigLoaded();	Set to true if a previously saved configuration file containing valid device information was loaded.
int GetDeviceType(); SetDeviceType(int type);	Gets and sets device type (OBU or RSU).
CString& GetLogStateStr(); int GetLoggingState() SetLoggingState(unsigned short val);	Gets and sets the logging state for a device.
unsigned short GetMessageCount(); CString& GetMsgCountStr(); SetMessageCount(unsigned short val);	Gets and sets the message count that is incremented by the sender and sent in the test header of over the air messages.
int GetRssi(); CString& GetRssiStr(); SetRssi(signed short val);	Gets and sets the RSSI for the device's last message.
unsigned short GetSenderId(); CString& GetSenderIdStr(); SetSenderId(unsigned short id);	Gets and sets user configured Sender ID.
unsigned char GetTxPower(); SetTxPower(unsigned char val);	Gets and sets the WRM TX power level.
IncMessageCount();	Increments the message count in case of local device. Not used for remote devices.
bool IsRsu();	Returns true if the object represents an RSU.
bool IsObu();	Returns true if the object represents an OBU.
PrintHeaders(ostream &output, CString prefix); PrintHeaders(ostream &output);	Prints headers when saving configuration or logging WaveDeviceInfo attributes. An optional prefix for custom formatting.
ResetDeviceDefaults();	Sets all values to zero.
friend ostream &operator<<(ostream&, const CWaveDeviceInfo&); friend istream &operator>>(istream&, CWaveDeviceInfo&);	Read/writes this object to a stream. Used for logging configuration to disk during testing & writing configuration to disk.
CLocationInfo m_LocationInfo;	Current GPS Location information for this vehicle or intersection (not stopping locations).
CSignalInfo m_SignalInfo;	If this device is an RSU, this object is the current signal information.

Method/Attribute	Description
CLocationInfo m_TestLocationInfo;	Test header version of our location.
CVehicleInfo m_VehicleInfo;	If this device is an OBU, this object represents the current vehicle information.
BYTE m_WrmMacAddr[2];	MAC Address of the WRM connected to the HD represented by this object.

4.3.1.15 Class CWrmControl

CWrmControl provides WRM transmit and receive capability, with a separate full time receive thread and an internal packet queue. This class also encapsulates a single use of the WAVEApi, to provide one point of control for all T9App dialogs needing access to the local WRM. This is the only object that makes WaveApi calls. See Table 4-19 for details.

Table 4-19. CWrmControl Methods and Attributes

Method/Attribute	Description
bool ConfigLoaded();	Returns whether or not loading configuration from disk file was successful, if false, a configuration file was not found.
int GetConfigFromWrm();	Gets configuration from WRM.
CWrmMsg& GetLastSent();	Gets a copy of the last message that was sent to the WaveApi, used to support logging.
CWrmMsg& GetNextRx();	Gets the oldest message received from the WRM from this object's internal message queue.
int GetRxCount();	Returns received message count.
GetWrmConfig(wrm_configuration_parameters_type *Config);	Gets the local WRM configuration.
bool InitRxThread();	Initializes and starts WRM message receive thread.
PrintHeaders(ostream &output);	Prints field headers for saving and logging this objects configurable attributes.
ResetWrm();	Resets WRM via WAVEAPI
static UINT rx_thread(LPVOID pParam);	Receive thread.
int SaveWrmConfig(wrm_configuration_parameters_type* Config);	Request the given configuration to be saved to WRM, via the WAVEApi
void SendUpdate(CWaveDeviceInfo& device);	Sends packets over the air.

Method/Attribute	Description
SetDestMacAddress(BYTE *Address);	Sets destination MAC address for over the air messages.
SetWrmConfig(wrm_configuration_parameters_type* Config);	Sets this objects internal WRM configuration structure with the passed in one.
StopRxThread();	Stops WRM message receive thread.
friend ostream &operator<<(ostream&, const CWrmControl&);	Saves and logs this object's configurable attributes.
friend istream &operator>>(istream&, CWrmControl&);	Saves and logs this object's configurable attributes.
CString m_DestIpAddress;	Current destination IP address, which is user configurable.

4.3.1.16 Class CWrmMsg

The CWrmMsg class encapsulates a sent or received WRM message. It creates an over the air formatted message from a given CWaveDeviceInfo object, or it creates a CWaveDeviceInfo object from a received over the air message. See Table 4-20 for details.

Table 4-20. CWrmMsg Methods and Attributes

Method/Attribute	Description
int ParseMessage();	Parses a received over the air message of either OBU or RSU format, and creates a valid CWaveDeviceInfo object (accessible with the m_Device attribute) if successful. Return value is the parsed size, a value of -1 indicates an error while parsing.
SetWrmIpAddr(unsigned long IpAddr);	Sets the IP address of the receiver of this message. This value is necessary for making calibration adjustments to the RSSI.
CWaveDeviceInfo m_Device;	A WAVE device info object created by a successful call to ParseMessage().
BYTE m_SourceIpString[20];	For a received CWrmMsg object, this is the sender's IP address, for sends, this value is undefined.
unsigned short m_Length;	Message size, including padding.
ULARGE_INTEGER m_TimeStamp;	For a received CWrmMsg object, this holds the parsed value of the timestamp encoded by the sender of this message. For transmit, this value is not used.

4.4 Data

4.4.1 Structures

This section describes the data structures that are common to T9App classes for control and message processing functions. Where commonality was identified in messages, these common structures were created to eliminate redundancy and reduce code size and processing overhead.

The GPS location structure (Table 4-21) is used in both OBU and RSU messages, and also in the message headers.

Table 4-21. GPS Location Structure

TGPS_LOCATION	
DWORD	Longitude;
DWORD	Latitude;
WORD	AltitudeMsb;
BYTE	AltitudeLsb:4;
BYTE	Reserved:4;

The test header structure (Table 4-22) contains the data that is prepended onto each RSU or OBU message.

Table 4-22. Test Header Structure

T_TEST_HEADER	
BYTE	LengthMsb;
BYTE	LengthLsb:4;
BYTE	GpsTimeMsb:4;
WORD	GpsTimeLsb;
GPS_LOCATION	Location;
WORD	Heading;
WORD	SenderId;
WORD	MessageCount;
BYTE	TxPower:5;
BYTE	LogEnabled:1;
BYTE	Reserved:2;
BYTE	OsTimeHi;
WORD	OsTimeMid;
DWORD	OsTimeLow;

The stopping location structure (Table 4-23) contains the data associated with each stopping location.

Table 4-23. Stopping Location Structure

T_TS_STOPPING_INFO	
GPS_LOCATION	Location;
WORD	Direction;
BYTE	SignalState;
WORD	SignalStateTimeLeft;
WORD	DurationOfYellowLight;

The common message structure contains the fields that are common to both the OBU and RSU message bodies (i.e., fields directly following the message header).

Table 4-24. Common Message Structure

T_COMMON_MSG_HEADER	
BYTE	MessageType;
BYTE	TemporaryId[2];
BYTE	PrecisionIndicator;
GPS_LOCATION	Location;
DWORD	UTCTimeMsb;
BYTE	UTCTimeLsb;

The OBU VTV message structure (Table 4-25) contains the data for the over the air message.

Table 4-25. OBU VTV Message

T_VTV_MSG	
COMMON_MSG_HEADER	CommonHeader;
WORD	Heading;
WORD	Speed;
BYTE	LatAccelMsb;
BYTE	LatAccelLsb:4;
BYTE	LongAccelMsb:4;
BYTE	LongAccelLsb;
WORD	YawRate;
BYTE	ThrottlePos;

T_VTV_MSG	
BYTE	BrakeInfo;
WORD	SteeringWheelAngle;
BYTE	SignalAndControlStatus;
BYTE	SystemHealth:4;
BYTE	Reserved:4;
BYTE	VehicleLengthMsb;
BYTE	VehicleLengthLsb:6;
BYTE	VehicleWidthMsb:2;
BYTE	VehicleWidthLsb;

The RSU TS message structure (Table 4-26) contains the data for the over the air message.

Table 4-26. RSU TS Message

T_TS_MSG	
COMMON_MSG_HEADER	CommonHeader;
TS_STOPPING_INFO	SI1Info;
TS_STOPPING_INFO	SI2Info;
TS_STOPPING_INFO	SI3Info;
TS_STOPPING_INFO	SI4Info;

4.4.2 Global Variables

This section describes the list of global scope variables and the rationale for their use.

4.4.2.1 T9App Globals

Table 4-27 lists the global variables/functions used by the T9App.

Table 4-27. T9 App Globals

Function	Rationale
Operator <<	Required to support streams overloading cleanly. Many T9App classes define global overloaded functions for streams operation. Since this function is called by methods of the C++ standard lib classes, they need to be global in scope.
Operator >>	"

4.4.2.2 WAVE API Globals

Table 4-28 lists global variables used by the WAVE API Interface implementation.

Table 4-28. WAVE API Globals

Variable	Rationale
AntennaFactorTable	The T9App reused code from the WAVE API Tester. The T9App left the code unchanged for consistency.
MaxTxPowerTable	"
FullInList	"

4.4.2.3 PCAN USB Globals

Table 4-29 lists global variables used by the PCAN USB.

Table 4-29. PCAN USB Globals

Function	Rationale
PCAN_Init g_CAN_Init;	Denso copied the implementation from Grid Connect PCAN example code, which used global scope variables.
PCAN_Close g_CAN_Close;	"
PCAN_Status g_CAN_Status;	"
PCAN_Write g_CAN_Write;	"
PCAN_Read g_CAN_Read;	"
PCAN_VersionInfo g_CAN_VersionInfo;	"
UnloadDLL	"
LoadDLL	"
check_err	"
GetFunctionAdress	"
HINSTANCE g_i_DLL;	"

4.5 Design Goals and Constraints

4.5.1 Design Methodology

The T9App is a 100% object oriented (OO) design, implemented in Microsoft C++.

4.5.2 MFC, Standard Library Usage

The T9App leverages the Microsoft Foundation Classes for its GUI elements, internal lists, strings, and multi-threading support. The T9App uses standard C/C++ libraries whenever possible. The T9App uses standard libraries for string operations, time/date functions, file access, formatting text, and synchronization.

4.5.3 ANSI Compliance

For code that doesn't use MFC support classes, the code is mostly ANSI C++ compliant, with the exception of meeting VSCC requirements that the ANSI C++ standard can't meet (e.g., high-precision timers, logging / displaying of 64bit data types).

4.5.4 Memory Management

The T9App uses automatic memory management to the extent possible. To avoid memory leaks or management problems, all classes were designed to include references to objects or the actual objects statically. Where memory is dynamically allocated, it is encapsulated in objects, and then control of these objects is passed to standard list classes, so the objects will be automatically destroyed under normal program termination.

4.5.5 Naming Conventions

The T9App Member variables, function parameters, and local variables were named to reflect the contents of the data. For example, the variable named `m_Width` in the *CVehicle* class will contain the width of the vehicle the object describes.

4.6 Modifications and Enhancements

This section provides guidance for making enhancements for some expected scenarios.

4.6.1 Changing Over the Air Formats

The *CWrmMsg* class performs over the air message creation and parsing. Modify the appropriate message structures, header or body data. These structures are:

4.6.1.1 *T_TEST_HEADER*

Modify this structure to add or change information in the common message header that is prepended to each over the air message. Modify the *CWrmMsg::CreateTestHeader()* and *CWrmMsg::ParseTestHeader()* to support parsing and creation.

4.6.1.2 T_COMMON_MSG_HEADER

Modify this structure to add or change information in the first fields that are identical in the OBU and RSU message (e.g., Message Type, Temporary ID). Modify the CWrmMsg::CreateCommonMsgHeader() and CWrmMsg::ParseCommonHeader() to support the modified fields.

4.6.1.3 T_VTV_MSG

Modify this structure to add or change information in the OBU V2V message. Modify the CWrmMsg::ParseVtvMessage() and CWrmMsg::CreateObuMsgFromDevInfo(). In addition, modifications to the CVehicleInfo member data may be needed.

4.6.1.4 T_TS_MSG

Modify this structure to add or change the information in the RSU TS message. Modify the CWrmMsg::CreateRsuMsgFromDevInfo() and CWrmMsg::ParseVtvMessage(). In addition, modifications to the CSignalInfo member data may be needed.

4.6.2 Changing Data Logging

To add or modify fields in the log records, modify the both record header and the record data. For example, to add the logging of a new reception parameter, modify the CTestLog::LogReceptionParametersLabels() method to add the new field name in the desired place, and then modify the CTestLog::LogReceptionParameters() method to log the new field each time a packet is received. Transmit and Receive packets all have the same system of labels and logging for each record type. (OBUTX, OBURX, RSUTX, RSURX).

4.6.3 Changing Test Types

The T9App currently supports 3 test types: run for n seconds, run for n transmits, and passive listener mode. Modify the CTestStatusDialog class, which has methods that check for test completion using the test type variable. If additional test types are defined, modify the CTestOptions class and the CTestOptionsGui Class.

4.6.4 Changing T9App Configuration Loading / Saving

Each class has its own global overloaded operators >> and << for loading and saving its configuration (for all classes that support loading/saving). To load/save additional class information, add these methods to the overloaded operator definitions to support this. Each class also has its own field labels function which must be updated so the field labels for the test log entries and configuration saving are consistent.

WaveDeviceInfo Example:

Change the stream output global overloaded function:

```
friend ostream &operator<<( ostream&, const CWaveDeviceInfo&)
```

Change the stream input global overloaded function:

```
friend istream &operator>>( istream&, CWaveDeviceInfo&);
```

Change the field labels function:

```
CWaveDeviceInfo::PrintHeaders
```

4.6.5 Changing CAN messages

To support additional or modified CAN messages, modify the CCanMsg and CVehicleInfo classes. The CCanMsg::UpdateVehicle method parses the CAN message ID, switches on this ID, and calls 1 of 3 different parsing functions, ProcessVelocityMessage, ProcessAccelMessage, and ProcessDevicesMessage. Modify the existing parsing function for changes, or add a parsing function for a new CAN message.

Each CAN message parser has a reference to a vehicle object that it updates with the contents of the CAN message. Modify the CVehicle class definition to modify or add storage variables if for the changed data. See the CVehicleInfo class description (Section 4.3.1.13) for details. If the OBU V2V message must also be modified, see Section 4.6.1.

5 Validation Results

5.1 Introduction

This chapter describes the Acceptance Test Plan (ATP) for the Task 9 Application (T9App) developed in accordance with the requirements in Chapter 2.

5.1.1 Scope

This Task 9 ATP defines the tests Denso conducted to verify compliance with the Task 9 requirements (Chapter 2), the Application Message Specification (Chapter 3), and the Traffic Signal Interface Specification (Section 6.4). The T9App also complies with the Wave Radio Module (WRM) Interface Specification [2]. The T9App re-uses the WRM interface software developed and verified as part of Task 6D. The interface requirements were not re-verified in this test. Denso submitted a test report with the results of these tests to the VSCC for approval.

5.2 Test Configurations

This section describes how to connect a host device to a WRM. It also describes the test setups and initialization procedures required by the tests specified in subsequent sections.

5.2.1 WAVE Radio Module Network Connection

Set up each Host Device (HD) and WRM as shown in Figure 5-1. Each HD and WRM must have a unique IP address. Configure the HD Internet Protocol (IP) address to 192.168.001.1xx, where xx is the WRM unit number assigned by Denso. Configure each HD with a Subnet Mask of 255.255.255.000.

5.2.2 Test Setups

5.2.2.1 Generic Test Setup

For all test setups, each HD is connected to its corresponding WRM using an Ethernet crossover cable. Each HD has the T9App and Ethereal software installed. Ethereal is a software network analyzer that may be downloaded at no charge from www.ethereal.com. The test procedures use Ethereal to verify the contents of the IP packets sent from the HD to the WRM. The WRM is powered using a wall power supply. Connect HD comm. port 1 to the VSCC supplied differential GPS (DGPS) using a standard serial cable. Configure the COM1 port settings to 9600 baud, no parity bit, 8 data bits, 1 stop bit. See Figure 5-1. For subsequent test setups, the crossover cable is not labeled, and the power supply is not shown for simplicity.

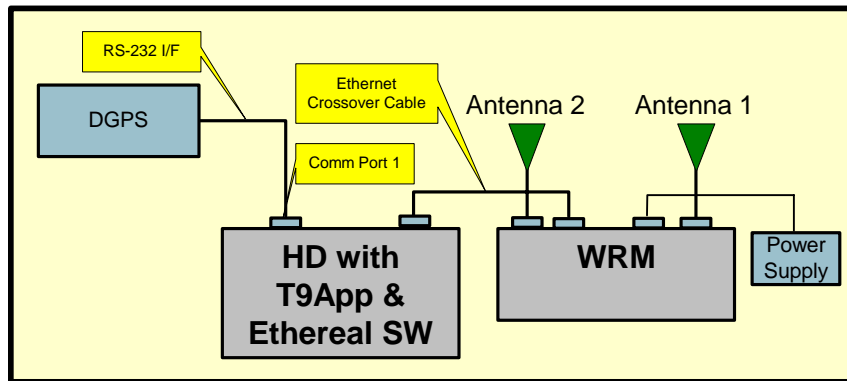


Figure 5-1. Generic Test Setup

5.2.2.2 OBU Test Setup

For OBU tests, install a Grid Connect Controller Area Network (CAN) bus adapter (see www.gridconnect.com for additional information) into the HD USB port and a second adapter into a laptop hosting the vehicle bus simulator (VBusSim) software. The VBusSim is a test application developed by Denso to verify the T9App. It enables the user to specify the CAN bus message contents. Connect the two adapters with a CAN bus cable. Verify the Peak System CAN (PCAN) software (supplied by Grid Connect) is installed on the HD. The test procedures use PCAN to verify the CAN bus message contents. See Figure 5-2.

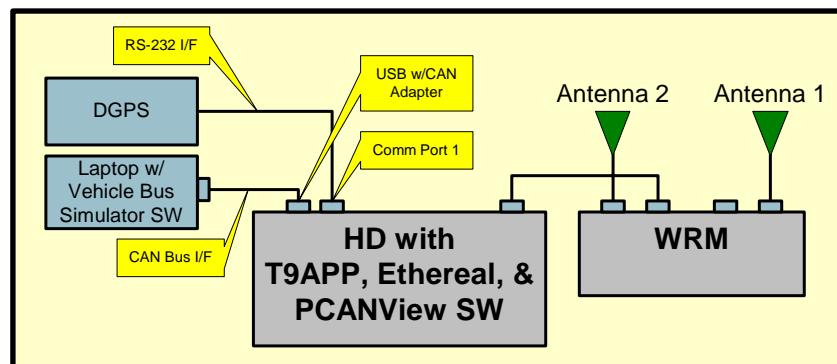


Figure 5-2. OBU Test Setup

5.2.2.3 RSU Test Setup

For RBU tests, connect HD comm port 2 to comm port 2 of a laptop hosting the traffic signal simulator (TrafSigSim) software using a standard serial cable. The TrafSigSim is a test application developed by Denso to verify the T9App. It responds to traffic signal queries and enables the user to specify the traffic signal message contents. Configure both comm ports to 19,200 bps, 8 data bits, no parity

bit, and 1 stop bit. The DGPS is not connected for some RSU tests. See Figure 5-3.

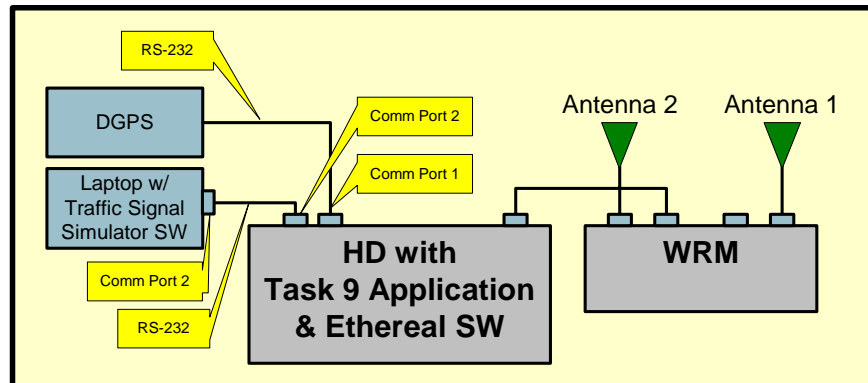


Figure 5-3. RSU Test Setup

5.2.3 Initialization

This section defines the procedures that are used to configure the hardware and software to a known state prior to starting a test. The tests in subsequent chapters refer to these procedures when required.

5.2.3.1 OBU and RSU Initialization Procedures

Table 5-1. OBU Initialization Procedure

#	Description	Test Steps	Expected Results
1.	Initialize HD.	If it is not already running, launch the T9App. If it is not already running, launch the Ethereal application.	The HD displays the T9App GUI and the Ethereal GUI.
	Initialize WRM.	On the T9App GUI, select WRM Configuration. On the WRM Configuration Screen, select Reset to WAVE default.	The WRM reboots and the T9App GUI displays the current WRM configuration.
	Initialize GUI parameters.	Initialize the T9App GUI parameters as described in Sections 5.2.3.2 and 5.2.3.3. Return to the main screen.	GUI displays entered values.
	Initialize GPS	If its not already running, turn on the GPS device and configure it up to output NMEA messages.	GPS device acquires satellite information and its GUI displays location information.
	Initialize VBusSim application.	If it is not already running, launch the VBusSim application on the connected laptop.	Laptop displays VBusSim GUI.

Table 5-2. RSU Initialization Procedure

#	Description	Test Steps	Expected Results
1.	Initialize HD.	If it is not already running, launch the T9App. If it is not already running, launch the Ethereal application.	The HD displays the T9App GUI and the Ethereal GUI.
	Initialize WRM.	On the TApp GUI, select WRM Configuration. On the WRM Configuration Screen, select Reset to WAVE default.	The WRM reboots and the Task 9 GUI displays the current WRM configuration.
	Set WRM to RSU mode.	On the WRM Configuration Screen, set the Unit Mode to RSU. Enter OK to return to the main screen. Select the WRM Configuration Screen. Confirm the unit mode is set to RSU. Enter OK to return to the main screen.	The T9App displays the unit mode as RSU.
	Initialize GUI parameters.	Initialize the GUI parameters as described in Sections 5.3, 5.4, and 5.5. Return to the main screen.	GUI displays entered values.
	Initialize GPS	If its not already running, turn on the GPS device and configure it up to output NMEA messages.	GPS device acquires satellite information and its GUI displays location information.
	Initialize Traffic Signal Simulator application.	If it is not already running, launch the Traffic Signal Simulator application on the connected laptop.	Laptop displays Traffic Signal Simulator GUI.

5.2.3.2 Communication Parameters Initialization

From the main screen, select Comm Parameters. On the Comm Parameters screen, set the parameters values as shown in Table 5-3.

Table 5-3. Communication Parameters Initial Values

Parameter	Value
Sender ID	Set to the last 3 digits of the host IP address.
Destination IP Address	255.255.255.255
Destination MAC Address	FF:FF:FF:FF:FF:FF
Message Interval (ms)	100
Message Size (bytes)	71 (OBU) or 125 (RSU)
GPS Group	Enabled (only if GPS is attached)
GPS Baud Rate	9600, N, 8, 1

Parameter	Value
GPS COM Port	COM1
Traffic Signal Setup	Enabled (RSU) or Disabled (OBU)
Traffic Signal Baud Rate	19200, N, 8, 1
Traffic Signal COM Port	COM2
Traffic Signal Polling Rate (ms)	500
CAN Setup	Enabled (OBU) or Disabled (RSU)
CAN Baud Rate	500 kbit/s
CAN Init Type	STD

5.2.3.3 Test Options Initialization

From the main screen, select Test Options. On the Test Options screen, set the parameter values as shown in Table 5-4.

Table 5-4. Test Options Initial Values

Parameter	Value
Test Control	Run for n seconds. Set seconds to 1000.
Logging	Enabled (select check box).
Test Name	ATP
Log Directory	C:\ATP

5.2.3.4 Traffic Signal Information Initialization

From the main screen, select Traffic Signal Information. On the Traffic Signal screen, set the Intersection # of Lanes to a value of four. Set the remaining parameters values as shown in Table 5-5.

Table 5-5. Traffic Signal Parameters Initial Values

Group	Latitude		Longitude		Altitude	Direction	Yellow Duration	Signal Phase
	Degrees	Direction	Degrees	Direction				
Intersection	90	N	180	E	-1000.0	0.0	N/A	N/A
Stop Loc 1	45.99999999	N	90.99999999	E	500.01	90.99	0.00	1
Stop Loc 2	0.0000017	N	0.0000017	E	1000.25	180.0	20.001	3
Stop Loc 3	45.5	S	90.5	W	2500.50	-90.55	40.999	6
Stop Loc 4	90	S	180	W	4000.99	-179.99	65.535	8

5.2.4 Software Builds

Record the WRM software version and the T9App software version in Table 5-6.

Table 5-6. WRM and T9App Software Versions

WRM Software Version	T9App Software Version	WRM Unit Number	Tests
WAVE Radio Module Ver 1.0	Denso WAVEtest V0.1	4, 10, 2, 6	All

5.3 GUI Parameter Tests

5.3.1 WRM Configuration Screen

1. Set up a RSU configuration as described in Section 5.2 (without the GPS receiver connected) and execute the RSU initialization procedure in Table 5-2.
2. Set up an OBU configuration as described in Section 5.2 and execute the OBU initialization procedure in Table 5-1.
3. On both the RSU and OBU WRM Configuration Screen, set the Tx power to 0 dBm, and record the WRM MAC address in Table 5-7.
4. Start an Ethereal session on both the RSU and OBU to log Ethernet activity sent from the HDs to the WRMs.
5. On the T9App main screen for both the RSU and OBU, select start testing. Record the RSSI value from the Test Display Screen in Table 5-7.
6. Allow the test to run for a minimum of 10 seconds, and then stop testing on both the RSU and OBU.
7. Stop recording Ethernet activity on the RSU and OBU Ethereal.
8. Open a Telnet session from the RSU and OBU HDs to the associated WRM. Use a Get RSSI Telnet command to get the RSSI of the last received packet. Verify these RSSI values match the RSSI values recorded from the GUIs.
9. Open the T9App log file on the RSU and OBU side. Verify the logged values match the values in Table 5-7 and record the results.
10. Use Ethereal on the RSU and OBU side to verify the IP data fields listed in Table 5-8 match the values in Table 5-7 and record the results.
11. Repeat steps 3-10 with power settings of 10 dBm, and full power, and record the results in Table 5-7.

Table 5-7. WRM Configuration Screen Results

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
RSU	WRM MAC Address	MAC Address	MAC Address	MAC Address	86:D9:FB:B1:26:1C	Pass	Pass
	RSSI (OBU Tx power setting = 0 dBm)	+/- 2 dB from RSSI Telnet value	+/- 2 dB from RSSI Telnet value	N/A	-55	-55	N/A
	<i>Telnet Get RSSI</i> (OBU Tx power setting = 0 dBm)	-55					
	OS Time (Message Timestamp)	N/A	Increments by ~ 100,000,000 in each message.	Increments by ~ 100,000,000 in each message.	N/A	Pass	Pass
	Message Type	N/A	0x01	0x01	N/A	0x01	0x01
	Temporary ID	N/A	RSU WRM MAC Address	RSU WRM MAC Address	N/A	Pass	Pass
	RSUTX Tx Power (setting = 0 dBm)	N/A	0	00000	N/A	0	Pass
	OBURX Tx Power (setting = 0 dBm)	N/A	0	00000	N/A	0	Pass
	RSSI (OBU Tx power setting = 10 dBm)	+/- 2 dB from RSSI Telnet value	+/- 2 dB from RSSI Telnet value	N/A	-44	-44	N/A
	<i>Telnet Get RSSI</i> (OBU Tx power setting = 10 dBm)	-45					
	RSUTX Tx Power (setting = 10 dBm)	N/A	10	01010	N/A	10	01010
	OBURX Tx Power (setting = 10 dBm)	N/A	10	01010	N/A	10	01010
	RSSI (OBU Tx power setting = full)	+/- 2 dB from RSSI Telnet value	+/- 2 dB from RSSI Telnet value	N/A	-37	-37	N/A

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
	<i>Telnet Get RSSI</i> (OBU Tx power setting = full)	-39					
	RSUTX Tx Power (setting = full)	N/A	31	11111	N/A	31	11111
	OBURX Tx Power (setting = full)	N/A	31	11111	N/A	31	11111
OBU	WRM MAC Address	MAC Address	MAC Address	MAC Address	B2:FE: 95:10: 1B:77	Pass	Pass
	RSSI (RSU Tx power setting = 0 dBm)	+/- 2 dB from RSSI Telnet value	+/- 2 dB from RSSI Telnet value	N/A	-59	-59	N/A
	<i>Telnet Get RSSI</i> (RSU Tx power setting = 0 dBm)	-57					
	OS Time (Message Timestamp)	N/A	Increment s by ~ 100,000,000 in each message.	Increment s by ~ 100,000,000 in each message.	N/A	Pass	Pass
	Message Type	N/A	0x00	0x00	N/A	0x00	0x00
	Temporary ID	N/A	OBU WRM MAC Address	OBU WRM MAC Address	N/A	Pass	Pass
	OBUTX Tx Power (setting = 0 dBm)	N/A	0	00000	N/A	0	00000
	RSURX Tx Power (setting = 0 dBm)	N/A	0	00000	N/A	0	00000
	RSSI (RSU Tx power setting = 10 dBm)	+/- 2 dB from RSSI Telnet value	+/- 2 dB from RSSI Telnet value	N/A	-47	-49	N/A
	<i>Telnet Get RSSI</i> (RSU Tx power setting = 10 dBm)	-48					
	OBUTX Tx Power (setting = 10 dBm)	N/A	0	01010	N/A	10	01010

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
	RSURX Tx Power (setting = 10 dBm)	N/A	0	01010	N/A	10	01010
	RSSI (RSU Tx power setting = full)	+/- 2 dB from RSSI Telnet value	+/- 2 dB from RSSI Telnet value	N/A	-37	-37	N/A
	<i>Telnet Get RSSI</i> (RSU Tx power setting = full)	-39					
	OBUTX Tx Power (setting = full)	N/A	31	11111	N/A	31	11111
	RSURX Tx Power (setting = full)	N/A	31	11111	N/A	31	11111

Table 5-8. WRM Configuration Data Elements

Data Element	Length (bits)	IP Data Frame Byte #	IP Data Frame Bits
OS Time	56	23-29	All
Message Type	8	30	1-8
Temporary ID	48	31-36	All
Tx Power of Sender	5	22	1-5

5.3.2 Comm Parameters Screen

1. Set up a RSU configuration as described in Section 5.2.2.3 (without the GPS receiver connected) and execute the RSU initialization procedure in Table 5-2.
2. Set up an OBU configuration as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1.

5.3.2.1 Comm Parameters Screen Low-Value Tests

1. On both the RSU and OBU Comm Parameters Screen, set the values as shown in Table 5-9 and enter OK.
2. Start an Ethereal session on both the RSU and OBU to log Ethernet activity sent from the HDs to the WRMs.
3. On the T9App main screen for both the RSU and OBU, select start testing.
4. Verify the GUI values match the values in Table 5-10 and record the results.
5. Allow the test to run for a minimum of 10 seconds, and then stop testing on both the RSU and OBU.

6. Stop recording Ethernet activity on both the RSU and OBU Ethereal.
7. Open the T9App log file on the RSU and OBU (Rx) side. Verify the logged values match the values in Table 5-10 and record the results.
8. Use Ethereal (on the RSU-side HD) to verify the RSU Traffic Signal Message IP data fields listed in Table 5-11 match the values in Table 5-10 and record the results. (**Note:** The Destination IP Address is part of the IP Frame, and the number of bytes for the Message Size must be counted.)

Table 5-9. Comm Parameters Screen Low Parameter Values

Parameter	Value
Destination IP Address	IP Address of other HD
Message Interval (msec)	10
Message Size (Bytes)	71 (OBU) or 125 (RSU)
Sender ID	0 (OBU) or 1 (RSU)

Table 5-10. Comm Parameters Screen Low Parameter Results

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
RSU	Destination IP Address	N/A	N/A	OBU IP Address	N/A	N/A	Pass
	Message Interval	N/A	~100 RSUTX entries/second	N/A	N/A	100 msgs in 1.046 seconds	N/A
	Packet Length	N/A	125 (RSUTX) 71 (OBU RX)	0x07D	N/A	125 71	0x07D 0x047
	Message Size	N/A	N/A	125 bytes	N/A	N/A	125
	Sender ID	1 (RSU) 0 (OBU)	1 (RSUTX), 0 (OBU RX)	0x0001	1 0	1 0	0x0001 0x0000
OBU	Destination IP Address	N/A	N/A	RSU IP Address	N/A	N/A	Pass
	Message Interval	N/A	~100 OBUTX entries/second	N/A	N/A	100 msgs in 1.044 seconds	N/A

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
	Packet Length	N/A	71 (OBUTX) 125 (RSURX)	0x047	N/A	71 125	0x047 0x07D
	Message Size	N/A	N/A	71 bytes	N/A	N/A	71
	Sender ID	0 (OBU) 1 (RSU)	0 (OBUTX) 1 (RSURX)	0x0000	0 1	0 1	0x0000 0x0001

Table 5-11. Comm Parameters Screen Data Elements

Data Element	Length (bits)	IP Data Frame Byte #	IP Data Frame Bits
Packet Length	12	1	1-8
		2	1-4
Sender ID	16	18-19	All

5.3.2.2 Comm Parameters Screen Mid-Value Tests

1. Repeat step 1 of Section 5.3.2.1, using the Comm Parameter Values in Table 5-12.
2. Repeat steps 2-8 of Section 5.3.2.1, verifying the values and recording the results in Table 5-13.

Table 5-12. Comm Parameters Screen Mid-Parameter Values

Parameter	Value
Destination IP Address	255.255.255.255
Message Interval (msec)	100
Message Size (Bytes)	500
Sender ID	255 (OBU) or 256 (RSU)

Table 5-13. Comm Parameters Screen Mid-Parameter Results

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
RSU	Destination IP Address	N/A	N/A	255.255. 255.255	N/A	N/A	255.255. 255.255
	Message Interval	N/A	~10 RSUTX entries/sec	N/A	N/A	10 msgs in 0.902 seconds	N/A
	Packet Length	N/A	500 (RSUTX) 500 (OBU RX)	0x1F4	N/A	500 500	0x1F4
	Message Size	N/A	N/A	500 bytes	N/A	N/A	500
	Sender ID	256 (RSU) 255 (OBU)	256 (RSUTX) 255 (OBU RX)	0x0100	256 255	256 255	0x0100 0x00FF
OBU	Destination IP Address	N/A	N/A	255.255. 255.255	N/A	N/A	255.255. 255.255
	Message Interval	N/A	~10 OBUTX entries/sec	N/A	N/A	10 msgs in 0.908 seconds	N/A
	Packet Length	N/A	500 (OBUTX) 500 (RSURX)	0x1F4	N/A	500 500	0x1F4
	Message Size	N/A	N/A	500 bytes	N/A	N/A	500
	Sender ID	255 (OBU) 256 (RSU)	255 (OBUTX) 256 (RSURX)	0x00FF	255 256	255 256	0x00FF 0x0100

5.3.2.3 Comm Parameters Screen High-Value Tests

1. Repeat step 1 of Section 5.3.2.1, using the Comm Parameter Values in Table 5-14.
2. Repeat steps 2-8 of Section 5.3.2.1, verifying the values and recording the results in Table 5-15.

Table 5-14. Comm Parameters Screen High-Parameter Values

Parameter	Value
Destination IP Address	255.255.255.255
Message Interval (msec)	1000
Message Size (Bytes)	1200
Sender ID	65534 (OBU) or 65535 (RSU)

Table 5-15. Comm Parameters Screen High-Parameter Results

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
RSU	Destination IP Address	N/A	N/A	255.255. 255.255	N/A	N/A	255.255. 255.255
	Message Interval	N/A	~1 RSUTX entry/sec	N/A	N/A	1.000 msgs/sec	N/A
	Packet Length	N/A	1200 (RSUTX) 1200 (OBURX)	0x4B0	N/A	1200 1200	0x4B0
	Message Size	N/A	N/A	1200 bytes	N/A	N/A	1200
	Sender ID	65535 (RSU) 65534 (OBU)	65535 (RSUTX) 65534 (OBURX)	0xFFFF	65535 65534	65535 65534	0xFFFF 0xFFFE
OBU	Destination IP Address	N/A	N/A	255.255. 255.255	N/A	N/A	255.255. 255.255
	Message Interval	N/A	~1 OBUTX entry/sec	N/A	N/A	1.000 msgs/sec	N/A
	Packet Length	N/A	1200 (OBUTX) 1200 (RSURX)	0x4B0	N/A	1200 1200	0x4B0
	Message Size	N/A	N/A	1200 bytes	N/A	N/A	1200
	Sender ID	65534 (OBU) 65535 (RSU)	65534 (OBUTX) 65535 (RSURX)	0xFFFE	65534 65535	65534 65535	0xFFFE 0xFFFF

5.3.3 Test Options Screen

1. Set up a RSU configuration as described in Section 5.2.2.3 (without the GPS receiver connected) and execute the RSU initialization procedure in Table 5-2.
2. Set up an OBU configuration as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1.
3. On both the RSU and OBU Test Options Screen, set the values as shown in Case 1 of Table 5-16 and enter OK.
4. Start an Ethereal session to log Ethernet activity between the RSU HD and the WRM.
5. On the T9App main screen for both the RSU and OBU, select start testing.
6. While the test is running, hit the Pause button on both the RSU and OBU.
7. Verify the message counts stop incrementing on both the RSU and OBU GUI.
8. Verify Ethereal is not detecting messages being sent or received.
9. Once the test has been paused for at least 5 seconds, hit the Resume button on the RSU and OBU.
10. Verify the GUI values match the values Table 5-17 and record the results.
11. For Test Case 1 and 2, wait until the test completes. For Test Case 3, wait 80 seconds and then hit the Quit button on both the RSU and OBU.
12. Stop recording Ethernet activity on Ethereal.
13. Open the T9App log file on the RSU and OBU side. Verify the logged values match the values in Table 5-17 and record the results.
14. Use Ethereal to view the IP data fields listed in Table 5-18 and verify the fields match the values in Table 5-17 and record the results.
15. Repeat steps 3-11 for Cases 2 and 3 in Table 5-16 and record the results in Table 5-19 and Table 5-20.

Table 5-16. Test Options Screen Parameter Values

Parameter	Values		
	Case 1	Case 2	Case 3
Test Control	Run for n sent messages. Set # of messages to 1000.	Run for n seconds. Set # of seconds to 50.	Run for n seconds. Set # of seconds to 100.
Logging	Enabled	Enabled	Disabled
Test Name	ATP	ATP	N/A
Log Directory	C:\ATP	C:\ATP	N/A

Table 5-17. Test Options Screen Case 1 Results

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
RSU	Message Count (test running)	Increments from 1 to # of messages sent or received.	RSUTX and OBURX increments from 1 to # of messages sent or received.	Increments from 1 to # of messages sent	Pass	Pass	Pass
	Message Count (test paused)	Message counts stop incrementing.	N/A	No messages being sent.	Pass	N/A	Pass
	Test Run	Last test run # + 1.	N/A	N/A	Pass	N/A	N/A
	Messages Sent	1000	N/A	N/A	Pass	N/A	N/A
	Seconds elapsed	~100	N/A	N/A	99	N/A	N/A
	Log	N/A	1	1	N/A	1	1
	Logfile name	N/A	c:\ATP\ATP_[test run]_mmddyy yy.txt	N/A	N/A	Pass	N/A
OBU	Message Count (test running)	Increments from 1 to # of messages sent or received.	OBUTX and RSURX increments from 1 to # of messages sent or received.	Increments from 1 to # of messages sent	Pass	Pass	Pass
	Message Count (test paused)	Message counts stop incrementing.	N/A	No messages being sent.	Pass	N/A	Pass
	Test Run	Last test run # + 1.	N/A	N/A	Pass	N/A	N/A
	Messages Sent	1000	N/A	N/A	Pass	N/A	N/A
	Seconds elapsed	~100	N/A	N/A	99	N/A	N/A
	Log	N/A	1	1	N/A	1	1
	Logfile name	N/A	c:\ATP\ATP_[test run]_mmddyy yy.txt	N/A	N/A	Pass	N/A

Table 5-18. Test Options Data Elements

Data Element	Length (bits)	IP Data Frame Byte #	IP Data Frame Bits
Message Count	16	20-21	All
Log	1	22	6

Table 5-19. Test Options Screen Case 2 Results

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
RSU	Message Count (test running)	Increments from 1 to # of messages sent or received.	RSUTX and OBURX increments from 1 to # of messages sent or received.	Increments from 1 to # of messages sent	Pass	Pass	Pass
	Message Count (test paused)	Message counts stop incrementing .	N/A	No messages being sent.	Pass	N/A	Pass
	Test Run	Last test run # + 1.	N/A	N/A	Pass	N/A	N/A
	Messages Sent	~500	N/A	N/A	499	N/A	N/A
	Seconds elapsed	50	N/A	N/A	50	N/A	N/A
	Log	N/A	1	1	N/A	1	1
	Logfile name	N/A	c:\ATP\ATP_[test run]_mmddyyy y.txt	N/A	N/A	Pass	N/A
OBU	Message Count (test running)	Increments from 1 to # of messages sent or received.	OBUTX and RSURX increments from 1 to # of messages sent or received.	Increments from 1 to # of messages sent	Pass	Pass	Pass
	Message Count (test paused)	Message counts stop incrementing .	N/A	No messages being sent.	Pass	N/A	Pass
	Test Run	Last test run # + 1.	N/A	N/A	Pass	N/A	N/A
	Messages Sent	~500	N/A	N/A	499	N/A	N/A
	Seconds elapsed	50	N/A	N/A	50	N/A	N/A
	Log	N/A	1	1	N/A	1	1

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
	Logfile name	N/A	c:\ATP\ATP_[test run]_mmddyyy y.txt	N/A	N/A	Pass	N/A

Table 5-20. Test Options Screen Case 3 Results

Unit	Parameter	Expected Results			Actual Results		
		GUI	Log	Ethereal	GUI	Log	Ethereal
RSU	Message Count	Increments from 1 to # of messages sent or received.	N/A	Increments from 1 to # of messages sent	Pass	N/A	Pass
	Test Run	"N/A"	N/A	N/A	"N/A"	N/A	N/A
	Messages Sent	~800	N/A	N/A	Pass	N/A	N/A
	Seconds elapsed	80	N/A	N/A	80	N/A	N/A
	Log	N/A	N/A	0	N/A	N/A	0
	Logfile name	N/A	No logfile generated	N/A	N/A	Pass	N/A
OBU	Message Count	Increments from 1 to # of messages sent or received.	N/A	Increments from 1 to # of messages sent	Pass	N/A	Pass
	Test Run	"N/A"	N/A	N/A	"N/A"	N/A	N/A
	Messages Sent	~800	N/A	N/A	Pass	N/A	N/A
	Seconds elapsed	80	N/A	N/A	Pass	N/A	N/A
	Log	N/A	N/A	0	N/A	N/A	0
	Logfile name	N/A	No logfile generated	N/A	N/A	Pass	N/A

5.3.4 Traffic Signal Information Screen

1. Set up a RSU configuration as described in Section 5.2.2.3 (without the GPS receiver connected) and execute the RSU initialization procedure in Table 5-2.
2. Set up an OBU configuration as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1.
3. Start an Ethereal session to log Ethernet activity between the RSU HD and the WRM.

4. On the T9App main screen for both the RSU and OBU, select start testing.
5. Verify the GUI values match the values in Table 5-21 for RSU (Tx) and OBU (Rx) sides, and record the results.
6. Allow the test to run for a minimum of 5 seconds, and then stop testing on both the RSU and OBU.
7. Stop recording Ethernet activity on Ethereal.
8. Open the T9App log file on the RSU (Tx) and OBU (Rx) side. Verify the logged values match the values in Table 5-21 and record the results.
9. Use Ethereal (on the RSU-side HD) to view the RSU Traffic Signal Message IP data fields listed in Table 5-22. Verify the fields match the values in Table 5-21 and record the results.

Table 5-21. Traffic Signal Information Screen Results

Group	Parameter	Expected Results			Actual Results				
		GUI	Log	Ethereal	Tx GUI	Tx Log	Ethereal	Rx GUI	Rx Log
Intersection	Latitude	90.0	90.0	0x35A4E900	90	90	Pass	90	90
	Longitude	180.0	180.0	0x6B49D200	180	180	Pass	180	180
	Altitude	N/A	-1000.0	0x00000	N/A	-1000	Pass	N/A	-1000
	Number of Lanes	N/A	4	N/A	N/A	4	N/A	N/A	N/A
Stop Loc 1	Latitude	N/A	45.9999999	0x1B6B0AFF	N/A	45.9999999	Pass	N/A	45.9999999
	Longitude	N/A	90.9999999	0x363D7F7F	N/A	90.9999999	Pass	N/A	90.9999999
	Altitude	N/A	500.01	0x249F1	N/A	500.01	Pass	N/A	500.01
	Direction	N/A	90.99	0x238B	N/A	90.99	Pass	N/A	90.99
	Yellow Duration	N/A	0.0	0x0000	N/A	0	Pass	N/A	0
Stop Loc 2	Latitude	N/A	0.0000017	0x00000011	N/A	0.0000017	Pass	N/A	0.0000017
	Longitude	N/A	0.0000017	0x00000011	N/A	0.0000017	Pass	N/A	0.0000017
	Altitude	N/A	1000.25	0x30D59	N/A	1000.25	Pass	N/A	1000.25
	Direction	N/A	180.0	0x4650	N/A	180	Pass	N/A	180
	Yellow Duration	N/A	20.001	0x4E21	N/A	20.001	Pass	N/A	20.001
Stop Loc 3	Latitude	N/A	-45.5	0xE4E14040	N/A	-45.5	Pass	N/A	-45.5
	Longitude	N/A	-90.5	0xCA0ECBC0	N/A	-90.5	Pass	N/A	-90.5
	Altitude	N/A	2500.50	0x55762	N/A	2500.5	Pass	N/A	2500.5

Group	Parameter	Expected Results			Actual Results				
		GUI	Log	Ethereal	Tx GUI	Tx Log	Ethereal	Rx GUI	Rx Log
	Direction	N/A	-90.55	0xDCA1	N/A	-90.55	Pass	N/A	-90.55
	Yellow Duration	N/A	40.999	0xA027	N/A	40.999	Pass	N/A	40.999
Stop Loc 4	Latitude	N/A	-90.0	0xCA5B1700	N/A	-90	Pass	N/A	-90
	Longitude	N/A	-180.0	0x94B62E00	N/A	-180	Pass	N/A	-180
	Altitude	N/A	4000.99	0x7A183	N/A	4000.99	Pass	N/A	4000.99
	Direction	N/A	-179.99	0xB9B1	N/A	-179.99	Pass	N/A	-179.99
	Yellow Duration	N/A	65.535	0xFFFF	N/A	65.535	Pass	N/A	65.535

Table 5-22. Traffic Signal Message Data Elements

Data Element		Length (bits)	IP Data Frame Byte #	IP Data Frame Bits
Intersection	Longitude	32	38-41	All
	Latitude	32	42-45	All
	Altitude	20	46-47	All
			48	1-4
Stop Loc 1	Longitude	32	54-57	All
	Latitude	32	58-61	All
	Altitude	20	62-63	All
			64	1-4
	Direction	16	65-66	All
	Yellow Duration	16	70-71	All
Stop Loc 2	Longitude	32	72-75	All
	Latitude	32	76-79	All
	Altitude	20	80-81	All
			82	1-4
	Direction	16	83-84	All
	Yellow Duration	16	88-89	All

Data Element		Length (bits)	IP Data Frame Byte #	IP Data Frame Bits
Stop Loc 3	Longitude	32	90-93	All
	Latitude	32	94-97	All
	Altitude	20	98-99	All
			100	1-4
	Direction	16	101-102	All
	Yellow Duration	16	106-107	All
Stop Loc 4	Longitude	32	108-111	All
	Latitude	32	112-115	All
	Altitude	20	116-117	All
			118	1-4
	Direction	16	119-120	All
	Yellow Duration	16	124-125	All

5.3.5 Test Display Screen

5.3.5.1 Distance Calculation, OBU Heading Test

1. Set up a RSU configuration as described in Section 5.2.2.3 (without the GPS receiver connected) and execute the RSU initialization procedure in Table 5-2.
2. Set up an OBU configuration as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1.
3. Record the GPS device readings in Table 5-23.
4. On the RSU Traffic Signal Parameters screen, set the intersection location values as shown in Case 1 of Table 5-24.
5. Calculate the distance between the RSU and OBU side by entering the GPS readings as the source, and the RSU Traffic Signal Intersection Location as the destination on the web site <http://jan.ucc.nau.edu/~cvm/latlongdist.html>.
6. On the T9App main screen for both the RSU and OBU, select start testing.
7. Verify the GUI values match the values in Table 5-25 and record the results.
8. Stop the test.
9. Open the T9App log file on the RSU and OBU side. Verify the logged values match the values in Table 5-25 and record the results.
10. Repeat steps 4-9 for Cases 2 and 3 in Table 5-24 and record the results in Table 5-25.

Table 5-23. Distance, Heading Test GPS Data

Parameter	GPS Device Readings
Longitude	-117.2275
Latitude	33.1333
Altitude	161.7 m

Table 5-24. Distance, Heading Test Parameter Values

Parameter Group	Parameter	Values		
		Case 1	Case 2	Case 3
RSU Traffic Signal Intersection Location	Latitude	Same as GPS latitude reading	GPS latitude reading + 0.0003 degrees	GPS latitude reading - 0.0015 degrees
	Longitude	Same as GPS longitude reading	GPS longitude reading + 0.0003 degrees	GPS longitude reading - 0.0015 degrees
	Altitude	Same as GPS altitude reading	Same as GPS altitude reading	Same as GPS altitude reading

Table 5-25. Distance, Heading Test Results

Test Case	Parameter	Expected Results		Actual Results			
		GUI	Log	RSU GUI	RSU Log	OBU GUI	OBU Log
1	Distance to Sender as calculated by web site	0					
	Distance to Sender	Same as web site +/- 5 meters.	Same as web site +/- 5 meters.	4.3 m	4.3 m	4.3m	4.3 m
	OBU Heading	Fluctuating Value	Fluctuating Value	Pass	Pass	N/A	N/A
2	Distance to Sender as calculated by web site	43.6 m					
	Distance to Sender	Same as web site +/- 5 meters.	Same as web site +/- 5 meters.	43.1 m	43.1 m	43.1 m	43.1 m
3	Distance to Sender as calculated by web site	217.8 m					
	Distance to Sender	Same as web site +/- 5 meters.	Same as web site +/- 5 meters.	218.5 m	218.5 m	218.5 m	218.5 m

5.3.5.2 RSU – RSU Bearing Test

1. Set up a RSU configuration as described in Section 5.2.2.3 (with the GPS receiver connected) and execute the RSU initialization procedure in Table 5-2.
2. Set up a second RSU configuration as described in Section 5.2.2.3 (without the GPS receiver connected) and execute the RSU initialization procedure in Table 5-2.
3. Record the GPS device readings in Table 5-26.
4. On the RSU #2 Traffic Signal Parameters screen, set the intersection location values as shown in Case 1 of Table 5-27.
5. Calculate the bearing between the RSUs using the Excel spreadsheet Task9Bearing.xls (spreadsheet generated by Denso). Enter the GPS readings for RSU #1, and the RSU Traffic Signal Intersection Location for RSU # 2 in Table 5-28.
6. On the T9App main screen for both the RSUs, select start testing.
7. Verify the GUI values match the values Table 5-28 and record the results.
8. Stop the test.
9. Open the T9App log file for both RSUs. Verify the logged values match the values in Table 5-28 and record the results.
10. Repeat steps 3-9 for Cases 2, 3, and 4 in Table 5-27 and record the results in Table 5-28.

Table 5-26. RSU – RSU Bearing Test GPS Data

Parameter	GPS Device Readings
Longitude	-117.2275
Latitude	33.1333
Altitude	161.7 m

Table 5-27. RSU – RSU Bearing Test Parameter Values

Parameter Group	Parameter	Value			
		Case 1	Case 2	Case 3	Case 4
RSU Traffic Signal Intersection Location	Latitude	GPS latitude reading + 0.0015 degrees	Same as GPS latitude reading	GPS latitude reading + 0.0015 degrees	GPS latitude reading -0.0015 degrees
	Longitude	Same as GPS longitude reading	GPS longitude reading + 0.0015 degrees	GPS longitude reading + 0.0015 degrees	GPS longitude reading -0.0015 degrees

Parameter Group	Parameter	Value			
		Case 1	Case 2	Case 3	Case 4
	Altitude	Same as GPS altitude reading	Same as GPS altitude reading	Same as GPS altitude reading	Same as GPS altitude reading

Table 5-28. RSU – RSU Bearing Test Results

Test Case	Parameter	Expected Results		Actual Results	
		GUI	Log	GUI	Log
1	Bearing from RSU #1 to RSU #2 calculated by spreadsheet.		0		
	Bearing from RSU #2 to RSU #1 calculated by spreadsheet.		180		
	RSU # 1 display/log	Same as spreadsheet +/- 2 degrees.	Same as spreadsheet +/- 2 degrees.	-0.7	-0.7
	RSU # 2 display/log	Same as spreadsheet +/- 2 degrees.	Same as spreadsheet +/- 2 degrees.	179.3	179.3
2	Bearing from RSU #1 to RSU #2 calculated by spreadsheet.		-90		
	Bearing from RSU #2 to RSU #1 calculated by spreadsheet.		90		
	RSU # 1 display/log	Same as spreadsheet +/- 2 degrees.	Same as spreadsheet +/- 2 degrees.	-91.4	-91.4
	RSU # 2 display/log	Same as spreadsheet +/- 2 degrees.	Same as spreadsheet +/- 2 degrees.	88.6	88.6
3	Bearing from RSU #1 to RSU #2 calculated by spreadsheet.		-39.9		
	Bearing from RSU #2 to RSU #1 calculated by spreadsheet.		140.1		
	RSU # 1 display/log	Same as spreadsheet +/- 2 degrees.	Same as spreadsheet +/- 2 degrees.	-41.1	-41.1
	RSU # 2 display/log	Same as spreadsheet +/- 2 degrees.	Same as spreadsheet +/- 2 degrees.	138.9	138.9
4	Bearing from RSU #1 to RSU #2 calculated by spreadsheet.		140.1		

Test Case	Parameter	Expected Results		Actual Results	
		GUI	Log	GUI	Log
	Bearing from RSU #1 to RSU #2 calculated by spreadsheet.	-39.9			
	RSU # 1 display/log	Same as spreadsheet +/- 2 degrees.	Same as spreadsheet +/- 2 degrees.	141.2	141.2
	RSU # 2 display/log	Same as spreadsheet +/- 2 degrees.	Same as spreadsheet +/- 2 degrees.	-38.8	-38.8

5.3.5.3 OBU – OBU Relative Heading Test

1. Set up an OBU configuration (with a GPS connected) as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1.
2. Set up a second OBU configuration (without a GPS connected) as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1.
3. On the OBU # 1 VBusSim GUI, set the CAN bus message parameters as shown in Case 1 of Table 5-29.
4. On the T9App main screen for both OBUs, select start testing.
5. Verify the OBU # 1 GUI values match the values in Table 5-30 and record the results.
6. Stop the test.
7. Repeat steps 3-6 for Cases 2 and 3 in Table 5-29 and record the results in Table 5-30.

Table 5-29. OBU – OBU Relative Heading Test Parameters

Parameter Group	Parameter	Values		
		Case 1	Case 2	Case 3
OBU Vehicle Velocity Message	Vehicle Speed	5.00 m/s	5.01 m/s	400.00 m/s
	Vehicle Speed Availability	1	1	1

Table 5-30. OBU – OBU Relative Heading Test Results

Test Case	Parameter	Expected Results	Actual Results
		GUI	GUI
1	OBU 1 relative heading to OBU # 2	0	"N/A"
2	OBU 1 relative heading to OBU # 2	Fluctuating Value	Fluctuating Value
3	OBU 1 relative heading to OBU # 2	Fluctuating Value	Fluctuating Value

5.3.6 GUI Configuration Recording, Persistence

1. Set up a RSU configuration as described in Section 5.2.2.3 (without the GPS receiver connected) and execute the RSU initialization procedure in Table 5-2.
2. Set up an OBU configuration as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1.
3. On the T9App main screen for both the RSU and OBU, select start testing.
4. Allow the test to run for a minimum of 5 seconds, and then stop testing on both the RSU and OBU.
5. Open the T9App log file on the RSU (Tx) and OBU (Rx) side. Verify the logged values in the Config entries match the values in Table 5-31 and record the results.
6. Close the T9App and cycle power on both the RSU and OBU HDs. Re-open the T9App on both the RSU and OBU.
7. Repeat steps 3-5 and record the results in Table 5-31.

Table 5-31. GUI Configuration Recording, Persistence Results

Test Step	Screen	Expected Results	Actual Results	
		Log	RSU Log	OBU Log
5	Communication Parameters	See Table 5-3.	Pass	Pass
	Test Options	See Table 5-4.	Pass	Pass
	Traffic Signal Parameters	See Table 5-5.	Pass	All zeros
7	Communication Parameters	See Table 5-3.	Pass	Pass
	Test Options	See Table 5-4.	Pass	Pass
	Traffic Signal Parameters	See Table 5-5.	Pass	All zeros

5.4 Interface Tests

5.4.1 GPS Receiver Interface Tests

5.4.1.1 Common Message Header – GPS Data Tests

1. Set up an OBU configuration as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1, without launching the VBusSim application.
2. Start the GPS and allow it to acquire satellite information and update its display.
3. Record the GPS device readings in Table 5-32.
4. Start an Ethereal session to log Ethernet activity between the HD and the WRM.
5. On the OBU T9App main screen, select start testing.
6. Allow the T9App test to run for at least 5 seconds.
7. Stop recording Ethernet activity on Ethereal.
8. Convert the time recorded on the GPS device to the number of seconds since the start of the week. Convert this value to hex to verify the “GPS seconds in week” value captured by Ethereal.
9. Use Ethereal to view the IP data fields indicated in Table 5-33 and record the values. Convert the hex values to decimal and verify the data corresponds to the GPS device values in Table 5-32. Record the results.

Table 5-32. Common Message Header GPS Data Test Results

Parameter	GPS Device Readings	Ethereal Results (decimal)
Longitude	-117.2275	-1163886040
Latitude	33.1333	331333293
Altitude	162.5m	116250
Time	21:24:50	249906

Table 5-33. Common Message Header GPS Data Elements

Data Element	Length (bits)	IP Data Frame Byte #	IP Data Frame Bits	Value (Hex)
GPS Seconds in Week	20	2	5-8	0x3
		3-4	All	0xD032
Longitude	32	5-8	All	0xBA208228
Latitude	32	9-12	All	0x13BFBEAD
Altitude	20	13-14	All	0x1C61
		15	1-4	0xA

5.4.1.2 OBU GPS Interface Tests

1. Set up a RSU configuration as described in Section 5.2.2.3, without the GPS receiver connected. Execute the RSU initialization procedure in Table 5-2, without launching Ethereal or TrafSigSim applications.
2. Set up an OBU configuration as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1, without launching the VBusSim application.
3. Start the GPS and allow it to acquire satellite information and update its display.
4. Record the GPS device readings in Table 5-34.
5. Start an Ethereal session to log Ethernet activity between the OBU HD and the OBU WRM.
6. On the T9App main screen for both RSU and OBU, select start testing.
7. Allow the test to run for at least 5 seconds.
8. Stop recording Ethernet activity on Ethereal.
9. Open the T9App log file on the OBU side and verify the logged GPS values match the values recorded by the GPS device. Verify the logged values in both the OBUTX entries and the RSURX log entries. In the OBUTX entries,

the GPS data is logged as the transmitted data. In the RXURX entries, the GPS data is logged as the receiver information. Record the results in Table 5-34.

10. Use Ethereal (on the OBU HD) to view the IP data fields indicated in Table 5-35 and record the values. Convert the hex values to decimal and verify the data corresponds to the GPS device values in Table 5-34. Record the results.
11. Convert the time recorded on the GPS device to the number of seconds since the start of the week. Use this calculated value to verify the OBU and RSU logged values for GPS seconds in the week. Convert this value to hex to verify the GPS seconds in the week value sent from the OBU, as captured by Ethereal.
12. Open the T9App log file on the RSU side and verify the logged GPS values in the OBURX entries match the values recorded by the GPS device. Record the results in Table 5-34.

Table 5-34. OBU GPS Interface Test Results

Parameter	GPS Device Readings	Actual Results			
		OBU Log		Ethereal	RSU Log OBURX
		OBUTX	RSURX		
Longitude	-117.2275	-117.2275	180	1172274637	-117.2275
Latitude	33.1333	33.1333	90	331333189	33.1333
Altitude	163.3 m	163.3	-1000	116330	163.3
Time	02:37:59	268679	268679	268679	268679

Table 5-35. OBU V2V Safety Message GPS Data Elements

Data Element	Length (bits)	IP Data Frame Byte #	IP Data Frame Bits	Value (Hex)
Longitude	32	38-41	All	0xBA20820F
Latitude	32	42-45	All	0x13BFBE45
Altitude	20	46-47	All	0x1C66
		48	1-4	0xA
# of seconds in week	40	2	5-8	0x4
		3-4	All	0x1987

5.4.1.3 RSU GPS Interface Tests

1. Set up a RSU configuration as described in Section 5.2.2.3 and execute the RSU initialization procedure in Table 5-2, without launching the Ethereal application.
2. Set up an OBU configuration as described in Section 5.2.2.2 without the GPS receiver connected. Execute the OBU initialization procedure in Table 5-1, without launching Ethereal or VBusSim applications.
3. Start the GPS and allow it to acquire satellite information and update its display.
4. Record the GPS device readings in Table 5-36.
5. Start an Ethereal session to log Ethernet activity between the RSU HD and the RSU WRM.
6. On the T9App main screen for both RSU and OBU, select start testing.
7. Review the T9App GUI on both the RSU and OBU to verify the RSU longitude and latitude values match the values recorded by the GPS device. Record the results in Table 5-36.
8. Allow the test to run for at least 5 seconds.
9. Stop recording Ethernet activity on Ethereal.
10. Open the T9App log file on the RSU side and verify the logged GPS values in both the RSUTX and OBURX entries match the values recorded by the GPS device. In the RSUTX entries, the GPS data is logged as the transmitted data. In the OBURX entries, the GPS data is logged as the receiver information. Record the results in Table 5-36.
11. Use Ethereal (on the RSU HD) to view the IP data fields indicated in Table 5-37 and record the values. Convert the hex values to decimal and verify the data corresponds to the GPS device values in Table 5-36. Record the results.
12. Convert the time recorded on the GPS device to the number of seconds since the start of the week. Use this calculated value to verify the OBU and RSU logged values for GPS seconds in the week. Convert this value to hex to verify the GPS seconds in the week value sent from the OBU, as captured by Ethereal.
13. Open the T9App log file on the OBU side and verify the logged GPS values in the RSURX entries match the values recorded by the GPS device. Record the results in Table 5-36.

Table 5-36. RSU GPS Interface Test Results

Parameter	GPS Device Readings	Actual Results				
		RSU GUI	RSU Log	Ethereal	RSU GUI	RSU Log (RSURX)
Longitude	-117.2275	-117.2275	-117.2275	1172274755	-117.2275	-117.2275
Latitude	33.1333	33.1333	33.1333	331333183	33.1333	33.1333
Altitude	163.3 m	N/A	166.6	117240	N/A	166.6
Time	02:57:00	N/A	269962	269962	N/A	269962

Table 5-37. RSU Traffic Signal Message GPS Data Elements

Data Element	Length (bits)	IP Data Frame Byte #	IP Data Frame Bits	Value (Hex)
Longitude	32	38-41	All	0xBA2081BD
Latitude	32	42-45	All	0x13BFBE3F
Altitude	20	46-47	All	0x1C9F
		48	1-4	0x8
# of seconds in week	40	2	5-8	0x4
		3-4	All	0x2383

5.4.2 Vehicle Bus Interface Tests

The tests in this section will verify low, mid and high values for all vehicle speed, yaw rate, lateral and longitudinal acceleration, throttle position, and steering wheel angle. The remaining discrete field values will also be verified.

1. Set up a RSU configuration as described in Section 5.2.2.3, without connecting the GPS receiver. Execute the RSU initialization procedure in Table 5-2 without launching the Ethereal or TrafSigSim applications.
2. Set up an OBU configuration as described in Section 5.2.2.2 (without the GPS receiver connected) and execute the OBU initialization procedure in Table 5-1.

5.4.2.1 Vehicle Bus Interface Low-Value Tests

1. Start an Ethereal session on the OBU HD to log Ethernet activity between the HD and the WRM.
2. Use the VBusSim to set the Vehicle Velocity settings as indicated in Table 5-38 and use PCAN to verify the Vehicle Velocity settings. Table 5-39 and use PCAN to verify the Vehicle Acceleration settings.
3. Use VBusSim to set the Vehicle Devices settings as indicated in Table 5-40 and use PCAN to verify the Vehicle Devices settings.

4. On the T9App main screen for both RSU and OBU, select start testing. Allow the test to run for a minimum of 10 seconds and then stop testing.
5. Stop recording Ethernet activity on Ethereal.
6. Review the T9App GUI on both the OBU and RSU to verify the displayed values match the expected GUI values in Table 5-41 and record the results.
7. Open the T9App log file on the OBU HD and verify the logged values in the OBUTX entries match the expected OBU log values in Table 5-41. Verify the vehicle speed is also logged in the RSURX entries as the receiver information. Record the results.
8. Use Ethereal (on the OBU HD) to view the IP data fields indicated in Table 5-42 and record the values in Table 5-41. Verify the Ethereal values match the expected values in Table 5-41.
9. Open the T9App log file on the RSU HD and verify the logged values in the OBURX entries match the expected RSU log values in Table 5-41. Record the results.

Table 5-38. Vehicle Velocity Settings (Low-Value)

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Vehicle Speed	0	1	0	1	0	1
Yaw Rate	-179.999	1	0xB9B1	1	0xB9B1	1

Table 5-39. Vehicle Acceleration Settings (Low-Value)

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Lateral Acceleration	-2.05	1	0xF7FE	1	0xF7FE	1
Longitudinal Acceleration	-2.05	1	0xF7FE	1	0xF7FE	1

Table 5-40. Vehicle Devices Settings (Low-Value)

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Throttle Position	0	1	0	1	0	1
Brake Applied Status	0	1	0	1	0	1
Brake Applied Pressure	0	1	0	1	0	1
Steering Wheel Angle	-179.98	1	0xB9B2	1	0xB9B2	1
Headlights	0	1	0	1	0	1
Turn Signal/Hazard Signal	0	1	0	1	0	1
Traction Control State	0	1	0	1	0	1
Anti-Lock Brake State	0	1	0	1	0	1
System Health	0	1	0	1	0	1

Table 5-41. Vehicle Interface Low-Value Test Results

Parameter	Expected Results			Actual Results				
	GUI	Log	Ethereal	OBU GUI	OBU Log	Ethereal	RSU GUI	RSU Log
Vehicle Speed	0 m/s	0	0		0	0	0	0
Yaw Rate	N/A	-179.99	0xB9B1	N/A	-179.99	0xb9b1	N/A	-179.99
Lateral Acceleration	N/A	-2.05	0xF33	N/A	-2.04	0xF34	N/A	-2.04
Longitudinal Acceleration	N/A	-2.05	0xF33	N/A	-2.04	0xF34	N/A	-2.04
Throttle Position	N/A	0	0	N/A	0	0	N/A	0
Brake Applied Status	N/A	0	0	N/A	0	0	N/A	0
Brake Applied Pressure	N/A	0	0	N/A	0	0	N/A	0
Steering Wheel Angle	N/A	-179.98	0xDCD9	N/A	-179.98	0xdcd9	N/A	-179.98
Headlights	N/A	0	0	N/A	0	0	N/A	0
Turn Signal/Hazard Signal	N/A	0	0	N/A	0	0	N/A	0
Traction Control State	N/A	0	0	N/A	0	0	N/A	0
Anti-Lock Brake State	N/A	0	0	N/A	0	0	N/A	0
System Health	N/A	0	0	N/A	0	0	N/A	0

Table 5-42. OBU V2V Safety Message Vehicle Data Elements

Data Element	Length (bits)	IP Data Frame Byte #	IP Data Frame Bits
Vehicle Speed	16	56	1-8
		57	1-8
Lateral Acceleration	12	58	1-8
		59	1-4
Longitudinal Acceleration	12	59	5-8
		60	1-8
Yaw Rate	16	61-62	All
Throttle Position	8	63	1-8
Brake Applied Status	4	64	5-8
Brake Applied Pressure	4	64	1-4
Steering Wheel Angle	16	65-66	All
Headlights	2	67	7-8
Turn Signal/Hazard Signal	2	67	5-6
Traction Control State	2	67	3-4
Anti-lock Brake State	2	67	1-2
System Health	4	68	1-4

5.4.2.2 Vehicle Bus Interface Mid-Value Tests

1. Repeat steps 1-6 of section 5.4.2.1, using the Vehicle Velocity settings in Table 5-43, the Vehicle Acceleration settings in Table 5-44, and the Vehicle Devices settings in Table 5-45.
2. Repeat steps 7-10 of section 5.4.2.1, verifying the values and recording the results in Table 5-46.

Table 5-43. Vehicle Velocity Settings (Mid-Value)

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Vehicle Speed	2.55	1	0x00FF	1	0x00FF	1
Yaw Rate	2.55	1	0x00FF	1	0x00FF	1

Table 5-44. Vehicle Acceleration Settings (Mid-Value)

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Lateral Acceleration	0	1	0x0000	1	0x0000	1
Longitudinal Acceleration	0	1	0x0000	1	0x0000	1

Table 5-45. Vehicle Devices Settings (Mid-Value)

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Throttle Position	7.5	1	0x0F	1	0x0F	1
Brake Applied Status	1	1	0x1	1	0x1	1
Brake Applied Pressure	7	1	0x7	1	0x7	1
Steering Wheel Angle	0	1	0x0000	1	0x0000	1
Headlights	1	1	0x1	1	0x1	1
Turn Signal/Hazard Signal	1	1	0x1	1	0x1	1
Traction Control State	1	1	0x1	1	0x1	1
Anti-Lock Brake State	1	1	0x1	1	0x1	1

Table 5-46. Vehicle Interface Mid-Value Test Results

Parameter	Expected Results			Actual Results				
	GUI	Log	Ethereal	Tx GUI	Tx Log	Ethereal	Rx GUI	Rx Log
Vehicle Speed	2.55 m/s	255	0x00FF	2.55	2.55	00FF	2.55	2.55
Yaw Rate	2.55 m/s	255	0x00FF	N/A	2.55	00FF	N/A	2.55
Lateral Acceleration	N/A	0	0x000	N/A	0	000	N/A	0
Longitudinal Acceleration	N/A	0	0x000	N/A	0	000	N/A	0
Throttle Position	N/A	7.5	0x0F	N/A	7.5	0F	N/A	7.5
Brake Applied Status	N/A	1	0x1	N/A	1	1	N/A	1
Brake Applied Pressure	N/A	7	0x7	N/A	7	7	N/A	7
Steering Wheel Angle	N/A	0	0x0000	N/A	0	0000	N/A	0
Headlights	N/A	1	0x1	N/A	1	1	N/A	1
Turn Signal/Hazard Signal	N/A	1	0x1	N/A	1	1	N/A	1
Traction Control State	N/A	1	0x1	N/A	1	1	N/A	1
Anti-Lock Brake State	N/A	1	0x1	N/A	1	1	N/A	1

5.4.2.3 Vehicle Bus Interface High-Value Tests

1. Repeat steps 1-6 of section 5.4.2.1, using the Vehicle Velocity settings in Table 5-47, the Vehicle Acceleration settings in Table 5-48, and the Vehicle Devices settings in Table 5-49.
2. Repeat steps 7-10 of section 5.4.2.1, verifying the values and recording the results in Table 5-50.

Table 5-47. Vehicle Velocity Settings (High-Value)

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Vehicle Speed	655.35	1	0xFFFF	1	0xFFFF	1
Yaw Rate	179.99	1	0x464F	1	0x464F	1

Table 5-48. Vehicle Acceleration Settings (High-Value)

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Lateral Acceleration	2.04	1	0x07F8	1	0x07F8	1
Longitudinal Acceleration	2.04	1	0x07F8	1	0x07F8	1

Table 5-49. Vehicle Devices Settings (High-Value)

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Throttle Position	100	1	0xC8	1	0xC8	1
Brake Applied Status	2	1	0x2	1	0x2	1
Brake Applied Pressure	15	1	0xF	1	0xF	1
Steering Wheel Angle	179.98	1	0x464E	1	0x464E	1
Headlights	2	1	0x2	1	0x2	1
Turn Signal/Hazard Signal	2	1	0x2	1	0x2	1
Traction Control State	2	1	0x2	1	0x2	1
Anti-Lock Brake State	2	1	0x2	1	0x2	1

Table 5-50. Vehicle Interface High-Value Test Results

Parameter	Expected Results			Actual Results				
	GUI	Log	Ethereal	Tx GUI	Tx Log	Ethereal	Rx GUI	Rx Log
Vehicle Speed	655.35 m/s	655.35	0xFFFF	655.35	655.35	0xFFFF	655.35	655.35
Yaw Rate	N/A	179.99	0x464F	N/A	179.99	0x464F	N/A	179.99
Lateral Acceleration	N/A	2.047	0x0CC	N/A	2.04	0x0CC	N/A	2.04
Longitudinal Acceleration	N/A	2.047	0x0CC	N/A	2.04	0x0CC	N/A	2.04
Throttle Position	N/A	100	0xC8	N/A	100	0xC8	N/A	100
Brake Applied Status	N/A	2	0x2	N/A	2	0x2	N/A	2
Brake Applied Pressure	N/A	15	0xF	N/A	15	0xF	N/A	15

Parameter	Expected Results			Actual Results				
	GUI	Log	Ethereal	Tx GUI	Tx Log	Ethereal	Rx GUI	Rx Log
Steering Wheel Angle	N/A	179.98	0x2327	N/A	179.98	0x2327	N/A	179.98
Headlights	N/A	2	0x2	N/A	2	0x2	N/A	2
Turn Signal/Hazard Signal	N/A	2	0x2	N/A	2	0x2	N/A	2
Traction Control State	N/A	2	0x2	N/A	2	0x2	N/A	2
Anti-Lock Brake State	N/A	2	0x2	N/A	2	0x2	N/A	2

5.4.2.4 Vehicle Bus Interface Discrete-Value Tests

1. Repeat steps 1, 2, 5, & 6 of section 5.4.2.1, using the Vehicle Devices settings in Table 5-51.
2. Repeat steps 7-10 of section 5.4.2.1, verifying the values and recording the results in Table 5-52.

Table 5-51. Vehicle Devices Settings

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Brake Applied Status	4	1	0x4	1	0x4	1
Headlights	3	1	0x3	1	0x3	1
Turn Signal/Hazard Signal	3	1	0x3	1	0x3	1
Traction Control State	3	1	0x3	1	0x3	1
Anti-Lock Brake State	3	1	0x3	1	0x3	1

Table 5-52. Vehicle Interface Test Results

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Brake Applied Status	4	0x4	4	0x4	4
Headlights	3	0x3	3	0x3	3
Turn Signal/Hazard Signal	3	0x3	3	0x3	3
Traction Control State	3	0x3	3	0x3	3
Anti-Lock Brake State	3	0x3	3	0x3	3

3. Repeat steps 1, 2, 5, & 6 of section 5.4.2.1, using the Vehicle Devices setting in Table 5-53.
4. Repeat steps 7-10 of section 5.4.2.1, verifying the values and recording the results in Table 5-54.

Table 5-53. Vehicle Devices Settings

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Brake Applied Status	8	1	0x8	1	0x8	1

Table 5-54. Vehicle Interface Test Results

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Brake Applied Status	8	0x8	8	0x8	8

5. Repeat steps 1, 2, 5, and 6 of section 5.4.2.1, using the Vehicle Devices setting in Table 5-55.
6. Repeat steps 7-10 of section 5.4.2.1, verifying the values and recording the results in Table 5-56.

Table 5-55. Vehicle Devices Settings

Parameter	Setting	Availability Indicator	PCAN Results			
			Expected		Actual	
			Setting	Availability Indicator	Setting	Availability Indicator
Brake Applied Status	15	1	0xF	1	0xF	1

Table 5-56. Vehicle Interface Test Results

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Brake Applied Status	15	0xF	F	0xF	F

5.4.3 Traffic Signal Interface Tests

5.4.3.1 Initialization/Polling Test

1. Set up a RSU configuration as described in Section 5.2.2.3 and execute the RSU initialization procedure in Table 5-2.

Fast Polling Test

1. On the T9App Comm Parameters screen, set the traffic signal polling interval to 100 milliseconds.
2. On the T9App main screen, select start testing.
3. On the TrafSigSim screen, verify it receives the 22 byte initialization sequence:
04 41 30 31 05 10 02 55 25 01 01 10 03 17 51 04 61 30 31 05 10 30
4. Verify the TrafSigSim responds with 10 30 10 31 followed by the specified amount of other data.
5. Verify the TrafSigSim receives nothing for at least a full second.
6. Verify the TrafSigSim receives a periodic polling query of: 04 61 30 31 05 10 30, and responds with a traffic signal message (contents to be verified in other tests).
7. Stop the test on the T9App.

Table 5-57. Traffic Signal Long Initial Response, Fast Polling Results

	TrafSigSim Results	
Parameter	Expected	Actual
Initialization Sequence	See step 3.	Passed. Received expected results
Initialization Response	See step 4. (83 bytes of other data may follow.)	Passed. TrafSigSim responded with 10 30 10 31. No additional data followed the initialization response.
Delay between receiving initialization and polling sequence	≥ 1 second.	Passed. Delay ≈ 1 second
Polling query	See step 6.	Passed. Received periodic query
Polling interval	~ 100 milliseconds	Passed. Polling rate is ~ 100 ms.

Slow Polling Test

1. On the T9App Comm Parameters screen, set the traffic signal polling interval to 1000 milliseconds.
2. Repeat steps 2-7 of Section 5.4.3.1 and record the results in Table 5-58.
3. Reset the T9App traffic signal polling interval to 100 milliseconds.

Table 5-58. Short Initial Response, Slow Polling Results

Parameter	TrafSigSim Results	
	Expected	Actual
Initialization Sequence	See Section 5.4.3.1, step 3.	Passed.
Initialization Response	See Section 5.4.3.1, step 4 followed by 0 bytes of other data.	Passed. TrafSigSim responded with 10 30 10 31. No additional data followed the initialization response.
Delay between receiving initialization and polling sequence	≥ 1 second.	Passed. Delay ≈ 1 second
Polling query	See Section 5.4.3.1, step 6.	Passed. Received periodic query
Polling interval	~ 1000 milliseconds	Passed. Polling rate is ~ 1000 ms

5.4.3.2 Traffic Signal Message Processing

1. Set up a RSU configuration as described in Section 5.2.2.3 (without the GPS receiver connected) and execute the RSU initialization procedure in Table 5-2.
2. Set up an OBU configuration as described in Section 5.2.2.2 and execute the OBU initialization procedure in Table 5-1.

Traffic Signal Message, Active Phase 1 and 5

1. Configure the TrafSigSim parameters and verify the results as shown in Table 5-59.
2. Start an Ethereal session to log Ethernet activity between the HD and the WRM on the RSU HD.
3. On the T9App main screen for both the RSU and OBU, select start testing.
4. Allow the test to run for a minimum of 5 seconds, and then stop testing on both the RSU and OBU.
5. Stop recording Ethernet activity on Ethereal.
6. Open the T9App log file on the RSU (Tx) and OBU (Rx) side. Verify the logged values match the Log values in Table 5-60 and record the results.
7. Use Ethereal (on the RSU HD) to view the RSU Traffic Signal Message IP data fields listed in Table 5-61 and record the values in Table 5-60. Verify the Ethereal values match the expected values in Table 5-60.

Table 5-59. Traffic Signal Active Phase 1 and 5 Parameters

Parameter	Setting	TrafSig Results			
		Expected		Actual	
		Byte	Value	Byte	Value
State of active phase on ring 1	Green	5	0x00	5	0x00
State of active phase on ring 2	Green	25	0x00	25	0x00
Active Phases	1, 5	44	0x11	44	0x11
Seconds left in the currently active phase of ring 1	1	12	0x01	12	0x01
Seconds left in the currently active phase of ring 2	1	32	0x01	32	0x01

Table 5-60. Traffic Signal Active Phase 1 and 5 Results

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Current State of Traffic Light at Stop Loc 1 (Phase 1)	0	0x00	0	Byte 67 = 0x00	0
Time Left in Current State at Stop Loc 1	1.000	0x03E8	1	Byte 68/69 = 0x03e8	1
Current State of Traffic Light at Stop Loc 2 (Phase 3)	3	0x03	3	Byte 85 = 0x03	3
Time Left in Current State at Stop Loc 2	0	0x0000	0	Byte 86/87 = 0x0000	0
Current State of Traffic Light at Stop Loc 3 (Phase 6)	3	0x03	3	Byte 103 = 0x03	3
Time Left in Current State at Stop Loc 3	0	0x0000	0	Byte 104/105 = 0x0000	0
Current State of Traffic Light at Stop Loc 4 (Phase 8)	3	0x03	3	Byte 121 = 0x03	3
Time Left in Current State at Stop Loc 4	0	0x0000	0	Byte 122/123 = 0x0000	0

Table 5-61. RSU Traffic Signal Message Data Elements

Data Element	Length (bits)	IP Data Frame Byte #	IP Data Frame Bits
Current State of Traffic Light at Stop Loc 1	8	67	All
Time Left in Current State at Stop Loc 1	16	68-69	All
Current State of Traffic Light at Stop Loc 2	8	85	All
Time Left in Current State at Stop Loc 2	16	86-87	All
Current State of Traffic Light at Stop Loc 3	8	103	All

Data Element	Length (bits)	IP Data Frame Byte #	IP Data Frame Bits
Time Left in Current State at Stop Loc 3	16	104-105	All
Current State of Traffic Light at Stop Loc 4	8	121	All
Time Left in Current State at Stop Loc 4	16	122-123	All

Traffic Signal Message, Active Phase 2 and 6

1. Repeat step 1 of Section 5.4.3.2, using the Traffic Signal settings in Table 5-62.
2. Repeat steps 2-7 of Section 5.4.3.2, verifying the values and recording the results in Table 5-63.

Table 5-62. Traffic Signal Active Phase 2 and 6 Parameters

Parameter	Setting	TrafSig Results			
		Expected		Actual	
		Byte	Value	Byte	Value
State of active phase on ring 1	Yellow	5	0x02	5	0x02
State of active phase on ring 2	Yellow	25	0x02	25	0x02
Active Phases	2, 6	44	0x22	44	0x22
Seconds left in the currently active phase of ring 1.	20	12	0x14	12	0x14
Seconds left in the currently active phase of ring 2.	20	32	0x14	32	0x14

Table 5-63. Traffic Signal Active Phase 2 and 6 Results

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Current State of Traffic Light at Stop Loc 1 (Phase 1)	3	0x03	3	Byte 67 = 0x03	3
Time Left in Current State at Stop Loc 1	0	0x0000	0	Byte 68/69 = 0x0000	0
Current State of Traffic Light at Stop Loc 2 (Phase 3)	3	0x03	3	Byte 85 = 0x03	3
Time Left in Current State at Stop Loc 2	0	0x0000	0	Byte 86/87 = 0x0000	0
Current State of Traffic Light at Stop Loc 3 (Phase 6)	2	0x02	2	Byte 103 = 0x02	2
Time Left in Current State at Stop Loc 3	20.000	0x4E20	20	Byte 104/105 = 0x4E20	20

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Current State of Traffic Light at Stop Loc 4 (Phase 8)	3	0x03	3	Byte 121 = 0x03	3
Time Left in Current State at Stop Loc 4	0	0x0000	0	Byte 122/123 = 0x0000	0

Traffic Signal Message, Active Phase 3 and 7

1. Repeat step 1 of Section 5.4.3.2, using the Traffic Signal settings in Table 5-64.
2. Repeat steps 2-7 of Section 5.4.3.2, verifying the values and recording the results in Table 5-65.

Table 5-64. Traffic Signal Active Phase 3 and 7 Parameters

Parameter	Setting	TrafSig Results			
		Expected		Actual	
		Byte	Value	Byte	Value
State of active phase on ring 1	Green	5	0x00	5	0x00
State of active phase on ring 2	Green	25	0x00	25	0x00
Active Phases	3, 7	44	0x44	44	0x44
Seconds left in the currently active phase of ring 1.	40	12	0x28	12	0x28
Seconds left in the currently active phase of ring 2.	40	32	0x28	32	0x22

Table 5-65. Traffic Signal Active Phase 3 and 7 Results

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Current State of Traffic Light at Stop Loc 1 (Phase 1)	3	0x03	3	Byte 67= 0x03	3
Time Left in Current State at Stop Loc 1	0	0x0000	0	Byte 68/69 = 0x0000	0
Current State of Traffic Light at Stop Loc 2 (Phase 3)	0	0x00	0	Byte 85 = 0x00	0
Time Left in Current State at Stop Loc 2	40.000	0x9C40	40	Byte 86/87 = 0x9C40	40
Current State of Traffic Light at Stop Loc 3 (Phase 6)	3	0x03	3	Byte 103 = 0x03	3
Time Left in Current State at Stop Loc 3	0	0x0000	0	Byte 104/105 = 0x0000	0

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Current State of Traffic Light at Stop Loc 4 (Phase 8)	3	0x03	3	Byte 121 = 0x03	3
Time Left in Current State at Stop Loc 4	0	0x0000	0	Byte 122/123 = 0x0000	0

Traffic Signal Message, Active Phase 4 and 8

1. Repeat step 1 of Section 5.4.3.2, using the Traffic Signal settings in Table 5-66.
2. Repeat steps 2-7 of Section 5.4.3.2, verifying the values and recording the results in Table 5-67.

Table 5-66. Traffic Signal Active Phase 4 and 8 Parameters

Parameter	Setting	TrafSig Results			
		Expected		Actual	
		Byte	Value	Byte	Value
State of active phase on ring 1	Yellow	5	0x02	5	0x02
State of active phase on ring 2	Yellow	25	0x02	25	0x02
Active Phases	4,8	44	0x88	44	0x88
Seconds left in the currently active phase of ring 1.	65	12	0x41	12	0x41
Seconds left in the currently active phase of ring 2.	65	32	0x41	32	0x41

Table 5-67. Traffic Signal Active Phase 4 and 8 Results

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Current State of Traffic Light at Stop Loc 1 (Phase 1)	3	0x03	3	Byte 67= 0x03	3
Time Left in Current State at Stop Loc 1	0	0x0000	0	Byte 68/69 = 0x0000	0
Current State of Traffic Light at Stop Loc 2 (Phase 3)	3	0x03	3	Byte 85 = 0x03	3
Time Left in Current State at Stop Loc 2	0	0x0000	0	Byte 86/87 = 0x0000	0
Current State of Traffic Light at Stop Loc 3 (Phase 6)	3	0x03	3	Byte 103 = 0x03	3
Time Left in Current State at Stop Loc 3	0	0x0000	0	Byte 104/105 = 0x0000	0

Parameter	Expected Results		Actual Results		
	Log	Ethereal	Tx Log	Ethereal	Rx Log
Current State of Traffic Light at Stop Loc 4 (Phase 8)	65.000	0x00	2	Byte 121 = 0x02	2
Time Left in Current State at Stop Loc 4	0	0x0FDE8	65	Byte 122/123 = 0xFDE8	65

5.5 Requirement/Test Cross Reference Matrix

5.5.1 Task 9 Software Requirement Verification

These are a summary of requirements gleaned from Chapter 2.

Table 5-68. Task 9 Software Requirement Cross Reference

Section Number	Requirement	Test Procedure
3	The Task 9 software shall have the capability to run one application at a time with the Task 9 device being configurable to easily choose amongst a set of preloaded applications. The On-board Unit (OBU) Test application will be used in a vehicle and will wirelessly broadcast vehicle parameters and decode incoming packets containing surrounding vehicle parameters. The Roadside Unit (RSU) Test application will be used near a roadway infrastructure device such as a traffic signal and will broadcast traffic signal information.	All tests, by inspection.
<i>3.1 OBU Application Requirements</i>		
3.1.1	In the OBU Test application, the Task 9 device shall use a Controller Area Network (CAN) bus to receive messages with vehicle information such as vehicle speed, brake position, acceleration, etc.	5.4.2
3.1.2	The OBU Test application shall periodically issue commands to the WRM to transmit the latest GPS and vehicle information wirelessly.	5.4.1.2
3.1.2	The periodicity of the transmitted information shall be adjustable through the control user interface and shall allow periods at least as small as 10 milliseconds.	5.3.2
<i>3.2 RSU Application Requirements</i>		
3.2.1	The RSU Test application shall decode traffic signal information from an RS-232 connection.	5.4.3
3.2.2	The RSU Test application shall periodically issue commands to the WRM to transmit traffic signal information wirelessly.	5.3.2
3.2.2	The periodicity of the transmitted information shall be adjustable through the control user interface and shall allow periods at least as small as 10 milliseconds.	5.3.2
3.2.3.1	The RSU Test application shall be configurable so that the user has the capability to modify constant parameters for the traffic signal information.	5.3.4

Section Number	Requirement	Test Procedure
3.3 OBU and RSU Test Common Requirements		
3.3.1	In the OBU Test application, the Task 9 device shall use an RS-232 port to receive ASCII messages formatted in the National Maritime Electronics Association (NMEA) 0183 standard. The application shall parse these messages to get the GPS data required in other parts of this document.	5.4.1.2
3.3.3	In addition to the data content sent from the OBU Test and RSU Test applications, each data packet shall also contain the following:	(N/A)
	<i>Sender ID</i> (either preset OR randomized at start of test)	5.3.2
	<i>Common Message ID</i> (to be defined by VSCC members)	5.3.1
	<i>Message Count</i> (incrementing short unsigned integer)	5.3.3
	<i>Broadcast Power</i>	5.3.1
	Operating System Time (nanoseconds)	5.3.1
	<i>State of Data Logging</i> (i.e., logging active, logging inactive)	5.3.3
	<i>Latitude, Longitude, Height</i> (ellipsoidal)	5.4.1.1
	<i>GPS Seconds in Week</i> (conversion from UTC time to GPS Seconds in Week)(set <i>GPS Seconds in Week</i> to zero for devices with no GPS receiver attached)	5.4.1.1
	<i>Heading</i> (from GPS receiver)	5.4.1.1
3.3.4	By default, the Task 9 device shall record all data sent and all data received from surrounding devices via the WRM.	5.3, 5.4
3.3.4	In addition, the Task 9 device shall record reception parameters, timing, and statistics associated with each data packet. The recorded reception parameters include:	(N/A)
	Receiver Signal Strength Indicator (RSSI)	5.3.1
	<i>Distance to Sender</i> (using valid GPS coordinates of sender and receiver)	5.3.5.1
	<i>Heading to Sender</i> (using valid GPS coordinates of sender and receiver)	5.3.5.3
	Current <i>GPS Seconds in Week</i> of receiver	5.4.1
	<i>Latitude, Longitude, Height</i> (ellipsoidal) of receiver	5.4.1
	<i>Heading</i> (from GPS receiver)	5.4.1
3.3.4	<i>Speed</i> (meters/second, via CAN from host vehicle if receiver)	5.4.2
	All parameters controllable from the control user interface shall be recorded.	5.3.6
3.3.5.1	The applications shall retain the value of all parameters controllable from the control user interface during a “normal” power cycle.	5.3.6
3.3.5.1	The control user interface shall allow the user to configure the applications to transmit wirelessly through the WRM upon application execution. This implies the application will also be decoding CAN and/or serial data as appropriate for the application.	5.3, 5.4
3.3.5.1	The control user interface shall allow the user to adjust all WRM communication parameters described in the WRM interface specification.	Tested in Task 6D
3.3.5.1	The control user interface shall display the IP address of the Task 9 device at the Ethernet port connected to the WRM.	5.3.2

Section Number	Requirement	Test Procedure
3.3.5.1	<p>The OBU Test and RSU Test applications shall each provide a display for configuring the communication parameters. The display shall allow the user to adjust the following communication parameters inherent to the application:</p> <p>Periodicity of message transmission</p> <p>Size of message being transmitted (append data to standard message using same message ID)</p> <p>Destination IP address (as a default, this should be the broadcast IP address)</p>	5.3.2
3.3.5.1	The OBU Test and RSU Test applications shall have the ability to initiate, suspend, or terminate a test. The test shall consist of the OBU Test or RSU Test applications transmitting and recording data as described in previous sections.	5.3.2
3.3.5.1	The length of the test shall be controllable via the control user interface. The user shall be able to control the length of the test by specifying the number of packets sent or the number of seconds of test time. During a test, the Task 9 device shall record data as outlined in sections above.	5.3.3
3.3.5.1	<p>The OBU Test and RSU Test applications shall have the ability to specify a filename for recorded test data. The format for the filename shall be <Test Name>_<Test Run>_<GPS Date> where:</p> <p><Test Name> is a field entered by the user</p> <p><Test Run> is an automatically incrementing number incremented after a test has ended (i.e., terminated by the user, maximum number off packet sent, test time exceeded)</p> <p><GPS Date> is derived from the local GPS information.</p>	5.3.3
3.3.5.1	The OBU Test and RSU Test applications shall each have an active visual display that shows other Task 9 devices that have communicated with the host Task 9 device. The display shall support at least 10 other Task 9 devices and update the information displayed at least once per second. The text on the display should be easily readable in a vehicular environment.	5.3.5
3.3.5.1	For each device, the following shall be displayed:	(N/A)
	<i>Sender ID</i>	5.3.2
	<i>State of Data Logging</i> (Requirement deleted during Denso/VSCC Telecon)	N/A
	<i>Most recent Message Count</i>	5.3.3
	<i>Distance to Sender</i>	5.3.5.1
	<i>Relative Heading to Sender = Heading to Sender</i> (from GPS differencing) – <i>Heading</i> (only to be calculated when host vehicle speed > 5 m/s)	5.3.5.3
	<i>RSSI</i>	5.3.1
	<i>Speed</i>	5.4.2

5.5.2 Common Message Header Verification

Table 5-69. Common Message Header Field Verification

Field Name	Verification Test
Packet Length	5.3.2
GPS Seconds in Week	5.4.1.1
GPS Longitude	5.4.1.1
GPS Latitude	5.4.1.1
GPS Altitude	5.4.1.1
GPS Heading	5.4.1.1
Sender ID	5.3.2
Message Count	5.3.2
Log	5.3.2
Tx Power	5.3.1
OS Time	5.3.1

5.5.3 OBU Message Verification

Table 5-70. OBU Message Verification

Field Name	Verification Test
Message Type	5.3.1
Temporary ID	5.3.1
Precision Indicator	N/A
Longitude of center of vehicle	5.4.1.2
Latitude of center of vehicle	5.4.1.2
Altitude of center of vehicle	5.4.1.2
UTC Time	5.4.1.2
Heading	5.4.1.2
Vehicle Speed	5.4.2
Lateral Acceleration	5.4.2
Longitudinal Acceleration	5.4.2
Yaw Rate	5.4.2
Throttle Position	5.4.2
Brake Applied Status	5.4.2
Brake Applied Pressure	5.4.2
Steering Wheel Angle	5.4.2

Field Name	Verification Test
Headlights	5.4.2
Turn Signal/Hazard Signal	5.4.2
Traction Control State	5.4.2
Anti-Lock Brake State	5.4.2
Vehicle Length	N/A
Vehicle Width	N/A

5.5.4 RSU Message Verification

Table 5-71. RSU Message Verification

Field Name	Verification Test
Message ID	5.3.1
Temporary ID	5.3.1
Precision Indicator	N/A
Longitude of RSU GPS Antenna	5.3.4, 0
Latitude of RSU GPS Antenna	5.3.4, 0
Altitude of RSU GPS Antenna	5.3.4, 0
UTC Time	5.4.1.3
Longitude of Stopping Location # 1	5.3.4
Latitude of Stopping Location # 1	5.3.4
Altitude of Stopping Location # 1	5.3.4
Directionality of Stopping Location # 1	5.3.4
Current State of Traffic Light at Stopping Location # 1	5.4.3.2
Time Left in Current State of Traffic Light at Stopping Location # 1	5.4.3.2
Duration of Yellow State at Stopping Location # 1	5.3.4
Longitude of Stopping Location # 2	5.3.4
Latitude of Stopping Location # 2	5.3.4
Altitude of Stopping Location # 2	5.3.4
Directionality of Stopping Location # 2	5.3.4
Current State of Traffic Light at Stopping Location # 2	5.4.3.2
Time Left in Current State of Traffic Light at Stopping Location # 2	5.4.3.2
Duration of Yellow State at Stopping Location # 2	5.3.4
Longitude of Stopping Location # 3	5.3.4
Latitude of Stopping Location # 3	5.3.4
Altitude of Stopping Location # 3	5.3.4
Directionality of Stopping Location # 3	5.3.4

Field Name	Verification Test
Current State of Traffic Light at Stopping Location # 3	5.4.3.2
Time Left in Current State of Traffic Light at Stopping Location # 3	5.4.3.2
Duration of Yellow State at Stopping Location # 3	5.3.4
Longitude of Stopping Location # 4	5.3.4
Latitude of Stopping Location # 4	5.3.4
Altitude of Stopping Location # 4	5.3.4
Directionality of Stopping Location # 4	5.3.4
Current State of Traffic Light at Stopping Location # 4	5.4.3.2
Time Left in Current State of Traffic Light at Stopping Location # 4	5.4.3.2
Duration of Yellow State at Stopping Location # 4	5.3.4

6 Appendix

6.1 Terms, Acronyms, and Abbreviations

Acronym	Definition
API	Application Programming Interface
ATP	Acceptance Test Plan
CAN	Controller Area Network
Comm	Communication
CTS	Clear to Send
DB	Decibel
DBi	Decibel Isotropic
DBm	Decibel Milliwatts
DBm	Decibel Milliwatts
Deg	Degrees
deg/sec	Degrees per Second
DGPS	Differential GPS
DLL	Dynamic Link Library
GPGLA	Global Positioning System Fix Data
GPRMC	Global Positioning Recommended Minimum Specific GPS/Transit Data
GPS	Global Positioning System
GPVTG	Global Positioning Track Made Good and Ground Speed
GPZDA	Global Positioning UTC Date / Time and Local Time Zone Offset
GUI	Graphical User Interface
HD	Host Device
I/F	Interface
IP	Internet Protocol
Kbps	Kilobits per Second
Lsb	Least Significant Bit
LSB	Least Significant Byte
LSBit	Least Significant Bit
LSByte	Least Significant Byte
LSNybble	Least Significant Nybble
M	Meters

Acronym	Definition
m/s	Meters per Second
MAC	Medium Access Control
Mbps	Megabits per Second
MFC	Microsoft Foundation Class
MHz	Megahertz
Ms	Milliseconds
MSB	Most Significant Byte
MSByte	Most Significant Byte
Msec	Milliseconds
N/A	Not Applicable
NMEA	National Maritime Electronics Association
OBU	On Board Unit
OBURX	On Board Unit Receive Log Entry
OBUTX	On Board Unit Transmit Log Entry
OS	Operating System
PCAN	Peak System Controller Area Network Software
PCAP	Packet Capture
RSSI	Received Signal Strength Indicator
RSU	Road Side Unit
RSURX	Road Side Unit Receive Log Entry
RTS	Request to Send
Rx	Receive
RXUTX	Road Side Unit Transmit Log Entry
SAE	Society of Automotive Engineers
Sec	Seconds
STD	Standard
Stop Loc	Stopping Location
T9App	Task 9 Application
TCP	Transport Control Protocol
TrafSigSim	Traffic Signal Simulator
Tx	Transmit
UML	Unified Modeling Language
USB	Universal Serial Bus
UTC	Coordinated Universal Time
UTC	Universal Coordinated Time

Acronym	Definition
V2V	Vehicle to Vehicle
VBusSim	Vehicle Bus Simulator
VSC	Vehicle Safety Communications
VSCC	Vehicle Safety Communications Consortium
WAVE	Wireless Access in Vehicular Environments
WRM	WAVE Radio Module

6.2 References

- [1] NMEA 0183 Interface Standard, dated January 2002.
- [2] Chapter 5, Vehicle Safety Communications Project, Task 6D: WAVE Radio Module, Final Task Report, September 27, 2004.

6.3 Task 9 device CAN input messages

This appendix outlines the CAN messages that the OBU Test application shall decode.

Parameters for the physical setup and messages will be:

- ISO 11898 Physical Layer
- 500 kilobits/second
- Standard 11-bit message IDs (not extended 29-bit)
- All signed values are twos-complement unless otherwise noted.

6.3.1 Vehicle Velocity Message

Message ID: 300 (hexadecimal)

Data Bytes: 8

Bit Byte	8	7	6	5	4	3	2	1(lsb)
1	Vehicle Speed MSByte							
2	Vehicle Speed LSByte (LSBit = 0.01 m/s; unsigned)							
3	Yaw Rate MSByte							
4	Yaw Rate LSByte (LSBit = 0.01 deg/sec; signed)							
5	Unused						Signal Availability Indicator (See Below)	

Byte 5:

Depending on Vehicle/Application some signals might not be available. A '1' indicates the signal is available and valid. A '0' indicates the signal is unavailable or otherwise invalid.

Bit	Signal
1	Vehicle Speed
2	Yaw Rate

6.3.2 Vehicle Acceleration Message

Message ID: 301 (hexadecimal)

Data Bytes: 8

Bit Byte	8	7	6	5	4	3	2	1(lsb)
1	Lateral Acceleration MSByte							
2	Lateral Acceleration LSByte (LSBit = 0.001 m/s; signed)							
3	Longitudinal Acceleration MSByte							
4	Longitudinal Acceleration LSByte (LSBit = 0.001 m/s; signed)							
5	Unused						Signal Availability Indicator (See Below)	

Byte 5:

Depending on Vehicle/Application some signals might not be available. A '1' indicates the signal is available and valid. A '0' indicates the signal is unavailable or otherwise invalid.

Bit	Signal
1	Lateral Acceleration
2	Longitudinal Acceleration

6.3.3 Vehicle Devices Message

Message ID: 302 (hexadecimal)

Data Bytes: 7

Bit Byte	8	7	6	5	4	3	2	1(lsb)
1	Throttle Position (LSBit = 0.5% open; unsigned)							
2	Brake Applied Status					Brake Applied Pressure		
3	Steering Wheel Angle MSByte							
4	Steering Wheel Angle LSByte (LSBit = 0.01 degrees; signed)							
5	Headlights (Meaning TBD)		Turn Signal/Hazard Signal (Meaning TBD)		Traction Control State (Meaning TBD)		Anti-Lock Brake State (Meaning TBD)	
6	Unused				System Health (Meaning TBD)			
7	Signal Availability Indicator (See Below)							

Byte 7:

Depending on Vehicle/Application some signals might not be available. A '1' indicates the signal is available and valid. A '0' indicates the signal is unavailable or otherwise invalid.

Bit	Signal
1	Throttle Position
2	Brake Applied Status
3	Steering Wheel Angle
4	Anti-Lock Brake State
5	Traction Control State
6	Turn Signal/Hazard Signal
7	Headlights
8	System Health

6.4 Traffic Signal Interface

This appendix specifies the format of the RS-232 messages that need to be exchanged in order to get traffic signal information into the Task 9 device.

Parameters for the physical setup and messages will be:

- 19,200 Baud
- 8 data bits
- No parity bit
- 1 stop bit

The communication between the Task 9 device and the traffic signal device shall consist of a series of requests and responses. The Task 9 device shall make requests and the traffic signal device will respond with data. There are two types of request/response pairs: initial and periodic.

6.4.1 Initial Request/Response State

The Task 9 device shall prepare the traffic signal controller for the periodic state by first transmitting the following sequence of 22 bytes (all bytes specified in hexadecimal format):

```
04 41 30 31 05 10 02 55 25 01 01 10 03 17 51 04
61 30 31 05 10 30
```

The Task 9 device shall then expect these four bytes to be transmitted from the traffic signal controller as a response:

```
10 30 10 31
```

There may also be an 83 byte data message following this response. These 83 bytes shall be ignored and discarded.

The Task 9 device shall then wait a full second before advancing to the periodic state.

6.4.2 Periodic Request/Response State

After leaving the "Initial State" the Task 9 device shall stay in the "Periodic State" for the duration of the program. The "Periodic State" shall consist of the Task 9 device repeatedly transmitting a request to the traffic signal device and the Task 9 device validating and parsing each response.

The Task 9 device shall execute the request by transmitting the following sequence of 7 bytes (all bytes specified in hexadecimal format) with a user-selectable period between 100 and 1000 milliseconds (with 100 milliseconds being the default):

```
04 61 30 31 05 10 30
```

The Task 9 device shall then expect the traffic signal controller to respond with a message that is at least 83 bytes long (with a maximum length of 100 bytes). Before validating and parsing the message it shall be searched for occurrences of the byte 0x10 in consecutive positions in the message. For every sequence of 0x10's one of them shall be extracted and discarded from the message. For example, if the received bytes were:

```
10 02 25 01 00 0B 1B 2A 2E 30 00 10 10 FF FF FF
FF FF FF FF FF 00 10 10 00 02 00 0B 1B 2A 2E 30
00 1A FF FF FF FF FF FF FF FF 00 1A 00 44 00 00
00 00 00 00 00 BB 00 00 00 00 00 00 00 00 00 00
00 44 00 00 FF 00 00 00 00 00 7F FF 00 00 00 00
10 03 2F 51 04
```

They would become this:

```
10 02 25 01 00 0B 1B 2A 2E 30 00 10 FF FF FF FF
FF FF FF FF 00 10 00 02 00 0B 1B 2A 2E 30 00 1A
FF FF FF FF FF FF FF FF 00 1A 00 44 00 00 00 00
00 00 00 BB 00 00 00 00 00 00 00 00 00 00 00 44
00 00 FF 00 00 00 00 00 7F FF 00 00 00 00 10 03
2F 51 04
```

after the "stuffed 0x10's" extraction.

Once this extraction has been performed the message shall be considered valid if it meets the following criteria:

1. It is exactly 83 bytes long.
2. It begins with the bytes 0x10 and 0x02 in that order.
3. It ends with the byte 0x04.

Once a message is determined to be valid then data shall be parsed from it. The following table shows the relevant bytes and their meanings:

Byte Number	Meaning
1	Header byte - constant 0x10
2	Header byte - constant 0x02
5	The current state of the active phase on ring 1: 0x00 = Green 0x02 = Yellow 0x03 = Red
25	The current state of the active phase on ring 2: 0x00 = Green 0x02 = Yellow 0x03 = Red
44	Active phases – one bit for each phase (1 – active, 0 – not active). Example: 0001 0001 – phases 1 and 5 are active 0010 0100 – phases 3 and 6 are active
46	Next active phases (format the same as "active phases").
12	Seconds left in the current state of the currently active phase of ring 1.
32	Seconds left in the current state of the currently active phase of ring 2.
83	End-of-message byte – constant 0x04

A **phase** can be thought of as a single flow of traffic through an intersection. An **active phase** is one that potentially has a non-red light. Any phase that is not active has a red light. The eight phases are divided into two groups of four phases called **rings**. Phases 1-4 make up ring 1 and phases 5-8 make up ring 2. Only 1 phase on each ring can be active simultaneously.

6.5 Preliminary Vehicle-to-Vehicle Common Message Set

Bit Byte	8	7	6	5	4	3	2	1(lsb)
1	Message Type (See Below)							
2	Temporary ID MSByte							
3	Temporary ID...							
4	Temporary ID...							
5	Temporary ID...							
6	Temporary ID...							
7	Temporary ID LSByte							
8	Precision Indicator (Meaning TBD)							
9	Longitude of center of vehicle MSByte							
10	"							
11	"							
12	Longitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)							
13	Latitude of center of vehicle MSByte							
14	"							
15	"							
16	Latitude LSByte (LSBit = 10 ⁻⁷ decimal degrees; signed)							
17	Altitude of center of vehicle MSByte							
18	Altitude (LSBit = 1 cm; unsigned; offset by +1 km)							
19	Unused				Altitude LSnybble			
20	UTC Time MSByte							
21								
22	Number of milliseconds since Jan. 1, 2004 at 00:00:00							
23								
24	UTC Time LSByte (LSBit = 0.001 seconds)							
25	Heading MSByte							
26	Heading (LSBit = 0.01 degrees; signed; 0 degrees = North)							
27	Vehicle Speed MSByte							
28	Vehicle Speed LSByte (LSBit = 0.01 m/s; unsigned)							
29	Lateral Acceleration MSByte							
30	Longitudinal Acceleration MSnybble				Lateral Acceleration LSnybble (LSBit = 0.01 m/s ² ; signed)			
31	Longitudinal Acceleration LSByte (LSBit = 0.01 m/s ² ; signed)							
32	Yaw Rate MSByte							
33	Yaw Rate LSByte (LSBit = 0.01 deg/sec; signed)							
34	Throttle Position (LSBit = 0.5% open; unsigned)							
35	Brake Applied Status (See Below)				Brake Applied Pressure (See Below)			
36	Steering Wheel Angle MSByte							
37	Steering Wheel Angle LSByte (LSBit = 0.02 degrees; signed)							
38	Headlights (See Below)	Turn Signal/Hazard Signal (See Below)			Traction Control State (See Below)		Anti-Lock Brake State (See Below)	
39	Unused				System Health (See Below)			
40	Vehicle Length MSByte							
41	Vehicle Width (Upper 2 bits)			Vehicle Length (Lower 6 bits; LSBit = 1 cm; unsigned)				
42	Vehicle Width LSByte (LSBit = 1 centimeter)							

NOTE: this represents the data format for the message set as delivered to the applications

MSByte - Most Significant Byte (8 bits)

LSByte - Least Significant Byte (8 bits)

LSNybble - Least Significant Nybble (4 bits)

All signed values are twos complement unless otherwise noted.

Message Type (From byte 1)	
0	Vehicle-to-Vehicle Message Version 1.0
1 - 255	Undefined

Headlights (From byte 38)	
00	Off
01	Daytime Running Lights
10	On
11	Brights

Brake Applied Status (From byte 35)	
0000	All Off
XXX1	Left Front
XX1X	Left Rear
X1XX	Right Front
1XXX	Right Rear
1111	All On

Turn Signal/Hazard Signal (From byte 38)	
00	Off
01	Left Turn Signal
10	Right Turn Signal
11	Hazard Signal

Brake Applied Pressure (From byte 35)	
0000	Not equipped
0001	Minimum braking pressure
0010	...
1111	Maximum braking pressure

Traction Control State (From byte 38)	
00	Not equipped
01	Off
10	On
11	Engaged

Anti-Lock Brake State (From byte 38)	
00	Not equipped
01	Off
10	On
11	Engaged

System Health (From byte 39)	
0000	No faults detected
0001	Specific error codes
...	"
1111	"

6.6 Task 9 Application Users' Guide

6.6.1 Introduction

This section provides the User's Guide for the Wireless Access in Vehicular Environments (WAVE) application delivered in accordance with the requirements in Chapter 2.

6.6.2 Scope

This User's Guide describes the setup, operating instructions, and troubleshooting procedures for the Task 9 Application (T9App). The T9App may be configured to operate in an OBU or RSU mode to support testing of vehicle safety scenarios.

6.6.3 Setup Procedures

6.6.3.1 Common Setup

This section describes the setup procedures that apply to both the OBU and RSU. It describes the WRM Setup, Host Device (HD) Configuration, Software Installation, and differential GPS (DGPS) setup. Figure 6-1 illustrates the configuration.

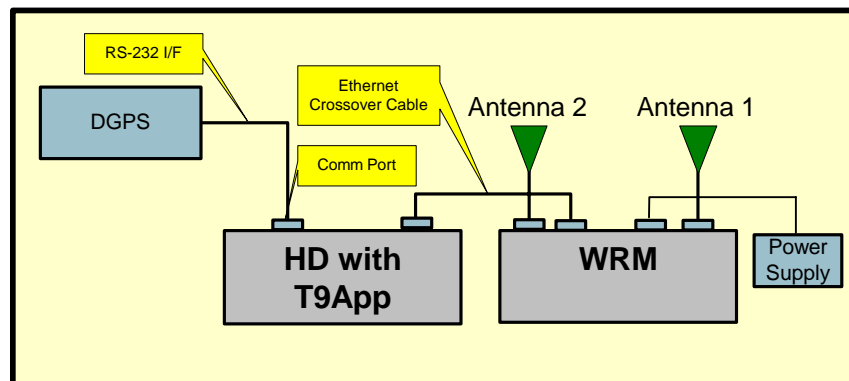


Figure 6-1. Common Setup

6.6.3.2 WRM Setup

Connect the HD to the WRM using one of the following methods:

- Use an Ethernet crossover cable to connect the HD Ethernet port directly to the WRM Ethernet port. If the HD is running Windows 2000 or Windows XP, a standard Ethernet cable may be used.

- Use standard Ethernet cables to connect the HD and WRM through a hub or Ethernet switch.

Supply power to the WRM using either the wall power supply (120V/5V AC-DC converter) or the vehicle power supply (12V/5V DC-DC converter with cigarette lighter adapter). For subsequent test setups, the crossover cable is not labeled, and the power supply is not shown for simplicity.

6.6.3.3 Host Device Configuration

Configure the HD IP address to an address of the format 192.168.001.xxx with Subnet Mask of 255.255.255.000. For communication to succeed, the HD IP address must be different from the WRM IP address (which is labeled on the WRM when it is delivered, and should not be changed by the user).

Each WRM has a unique IP address of the format 192.168.001.0xx, where xx is the DENSO assigned unit number. Each HD IP address needs to be unique as well. DENSO recommends adding 100 to the last set of digits in the WRM IP address to get the HD IP address. For example, if the WRM IP address is 192.168.001.013 then set the Host IP address to 192.168.001.113.

Windows IP Configuration Example

If the HD is a PC running the Windows XP operating system, use the following procedure to configure the HD IP address.

- From the Host PC's Start menu, choose Control Panel and then Network Connections.
- Right click on the Local Area Connection icon that belongs to the Ethernet controller connected to the WRM and select Properties.
- Within the Local Area Connection Properties dialog box, choose Internet Protocol (TCP/IP) and click Properties.
- Click the radio button corresponding to "Use the following IP address".
- Configure the Host IP address to (192.168.001.xxx) for the Ethernet connection in the Internet Protocol (TCP/IP) Properties dialog box. For example, if the WRM IP address is 192.168.001.013 then set the Host IP address to 192.168.001.113.
- Set the subnet mask to 255.255.255.0.
- Accept the settings and close the Internet Protocol Properties dialog box.
- The HD is now configured to communicate with the WRM.

6.6.3.4 Software Installation

Run the WAVETestInstall.exe program to install the T9App software. On the Customer Information screen, enter your User Name and Company Name. The installation places the program executable and dynamic link library files in the

appropriate directory (C:\Program Files\DENSO\WAVE Test\WAVE Test vx.x). The installation also places a WAVE Test vx.x (i.e., T9App) shortcut on your computer's Desktop. Double-clicking this icon will start the program. The program can also be started from the Windows Start button by selecting Programs and then WAVE Test vx.x.

The T9App program utilizes the Windows packet capture (PCAP) libraries, which must be installed separately. If the PCAP software was installed previously (e.g., as part of the API Tester installation), it does not need to be reinstalled. DENSO will supply the PCAP installation executable as part of the T9App delivery. It is also available from <http://winpcap.polito.it>.

6.6.3.5 GPS Setup

Connect a HD comm port to the DGPS using a standard serial cable (by default, the T9App uses comm port 1). Configure the comm port settings to the settings being used by the DGPS using the T9App Comm Parameters screen (see Section 6.6.4.4). The default T9App and CSI Wireless DGPS settings are 9600 baud, no parity bit, 8 data bits, and 1 stop bit. For the RSU, use of the DGPS is optional.

On some computers running Windows XP, the OS mistakenly interprets the DGPS as a serial track ball and loss of mouse control results. If this occurs, use the OS to disable the bogus track ball. From the Start menu, select Control Panel, and then System. Select the Hardware tab and then Device Manager. Right click on the device and disable it.

6.6.3.6 OBU Setup

Figure 6-2 illustrates the OBU setup. For the vehicle bus interface, install the Controller Area Network (CAN) bus drivers from the disk or CD supplied by Grid Connect (see www.gridconnect.com for additional information). Install a Grid Connect adapter into a HD USB port. VSCC must supply the cable to connect the adapter to the vehicle bus.

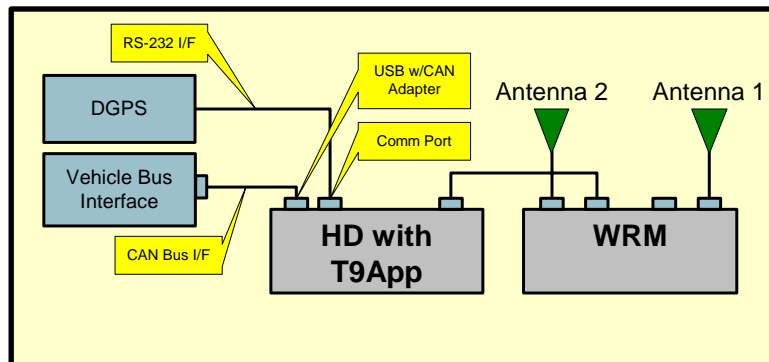


Figure 6-2. OBU Setup

6.6.3.7 RSU Setup

Figure 6-3 illustrates the RSU setup. Connect a HD comm port to the traffic signal controller using a standard serial cable (by default, the T9App uses comm port 2). Use the T9App Comm Parameters Screen (see Section 6.6.4.4) to configure the comm port settings. The default T9App settings are 19,200 baud, no parity bit, 8 data bits, and 1 stop bit.

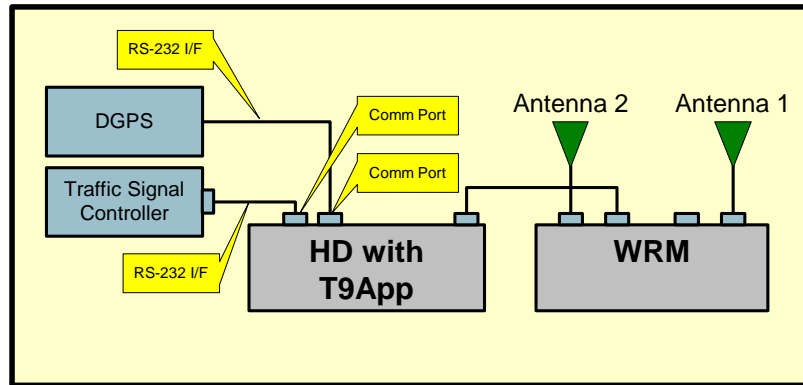


Figure 6-3. RSU Setup

6.6.4 Operating Instructions

6.6.4.1 Overview

The T9App provides the following functionality:

- User configuration of test parameters
- WRM configuration
- Transmission and reception of messages formatted in accordance with the Task 9 Application Message Specification defined in Section 6.5.
- Real-time display of WAVE network status
- Logging of configuration parameters and message traffic
- Operation in OBU or RSU mode
- Interface to DGPS receiver and CAN vehicle bus for OBU operation
- Interface to DGPS receiver and traffic signal controller for RSU operation

6.6.4.2 Main Screen

Figure 6-4 shows the T9App Main Screen Layout. It enables the user to access four screens (i.e., WRM Configuration, Comm Parameters, Test Options, and

Traffic Signal Information) to set up the test configuration. After setup, press the "Start Testing" button to begin test execution.

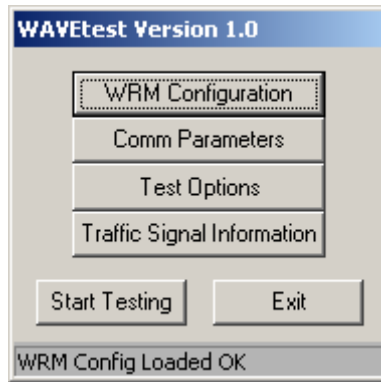


Figure 6-4. T9App Main Screen Layout

6.6.4.3 WRM Configuration Screen

The WRM Configuration Screen enables the user to configure the WRM parameters or reset the WRM configuration to the default settings. Figure 6-5 shows the WRM Configuration Screen layout. Refer to the WRM Interface Specification 0 for a detailed description of the parameter options and their default values. The unit mode setting specifies whether the T9App and WRM operate in RSU or OBU mode. The T9App limits the available options for Tx power to settings that are valid for the current channel, unit mode, and antenna compensation factor. Refer to the WRM Interface Specification for a description of the valid power settings. After selecting the desired settings, press the OK button to configure the WRM and exit the screen, or Cancel to leave the WRM configuration unchanged. Use the Get Current Configuration button to refresh the screen with the current WRM settings.

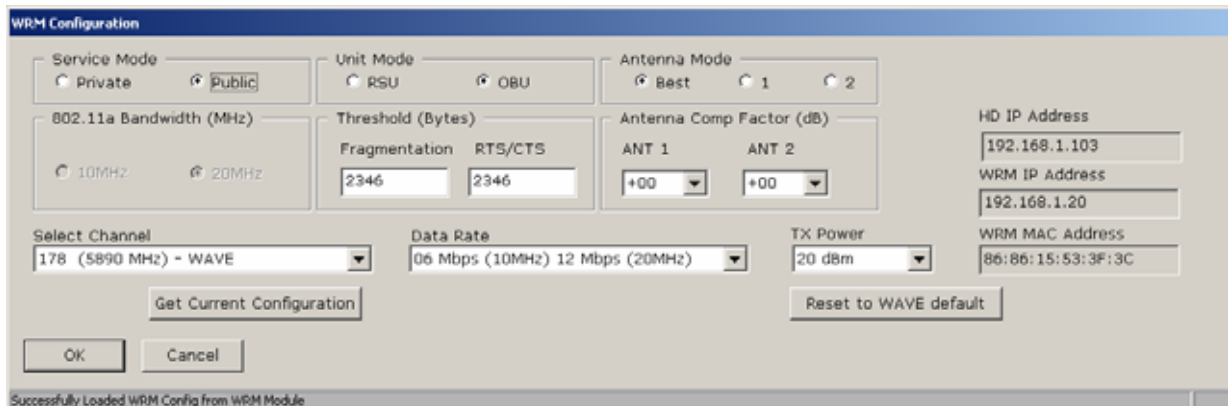


Figure 6-5. WRM Configuration Screen Layout

6.6.4.4 Comm Parameters Screen

The Comm Parameters Screen enables the user to configure communication parameters for over the air message transmission and device interfaces. Figure 6-6 shows the Comm Parameters Screen layout and Table 6-1. Comm Parameters Screen Field Definitions provides the field definitions.

Figure 6-6. Comm Parameters Screen Layout

Table 6-1. Comm Parameters Screen Field Definitions

Group	Field	Description
N/A	Local HD IP Address	See Section 6.6.3.3 for instructions on how to set the Host Device IP Address using OS capabilities.
	Sender ID	Set to the ID to be sent in the common message header.
	Destination HD IP Address	Set to 255.255.255.255 for broadcast IP addressing, or to the destination HD IP address for unicast IP addressing.
	Destination WRM MAC Address	Set to all FFs for broadcast MAC addressing, or to the destination WRM MAC address for unicast IP addressing. Note that OBUs automatically regenerate a new random MAC address whenever they are reset or power cycled.

Group	Field	Description
	Tx Message Interval	Set to the transmit interval for the OBU V2V Safety Message or the RSU Traffic Signal Message.
	Tx Message Size.	Set to the desired message size. The T9App adds fill bytes to the end of the message if needed to reach the specified size. The T9App ignores this parameter if it is less than the size of the OBU or RSU message.
GPS Setup	Enabled	Check the box if a GPS receiver is attached.
	Baud Rate	Set the baud rate to match the GPS Receiver configuration.
	Com Port	Set to the port on which the GPS Receiver is attached.
Traffic Signal Setup	Enabled	Check the box if a Traffic Signal Controller device is attached.
	Baud Rate	Set the baud rate to match the Traffic Signal Controller configuration.
	Com Port	Set to the port on which the Traffic Signal Controller is attached.
CAN Setup	Enabled	Check the box if a CAN bus is attached.
	Baud Rate	Set the baud rate to match the CAN bus configuration.
	Init Type	Set to standard (STD).

6.6.4.5 Test Options Screen

The Test Options Screen enables the user to configure the test duration and logging. Figure 6-7 shows the Test Options Screen layout and Table 6-2 provides the field definitions.

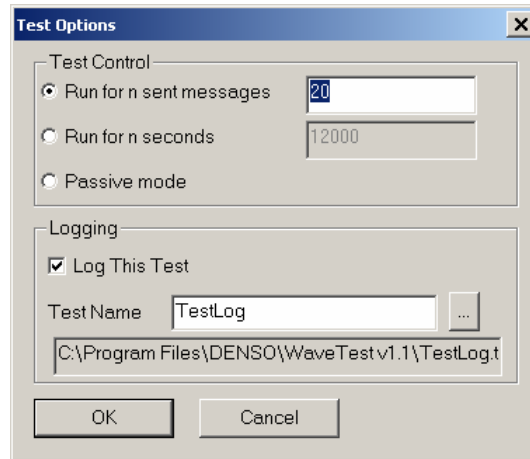


Figure 6-7. Test Options Screen Layout

Table 6-2. Test Option Screen Field Definitions

Group	Field	Description
Test Control	Run for n sent messages	Check the radio button and enter the total number of messages to be sent during the test.
	Run for n seconds	Check the radio button and enter the number of seconds the test should run.
	Passive mode	Check the radio button to run in passive mode. The T9App will not transmit any messages, but will display and log received messages. After starting the test, the test will continue to run until the user presses the Quit button.
Logging	Log this Test	Check the box to generate a logfile for the test. Uncheck the box if no logfile is required.
	Test Name	Enter the test name for the logfile. Browse to the desired directory by pressing the "...". The T9App will generate a logfile at the specified location with the name [Test Name]_[Test Run]_mmddyyyy.txt. The T9App will set the Test Run number to 1 for the first test of the day for this test name and location and automatically increment it for all subsequent tests.

6.6.4.6 Traffic Signal Information Screen

The Traffic Signal Information Screen enables the user to configure the static traffic signal information. Figure 6-8 shows the Traffic Signal Information screen layout and Figure 6-9 shows the Location Information dialog box that is displayed when any of the Location buttons are pushed. If the RSU has a GPS receiver attached, the RSU will use the GPS information as the intersection location rather than the user entered values. Table 6-3 provides the field definitions.

Traffic Signal Information

Intersection
Location

Number of Lanes: 0

Stopping Location 1
Location
Yellow light duration (sec): 5.500
Signal phase number: 1

Stopping Location 2
Location
Yellow light duration (sec): 10.000
Signal phase number: 2

Stopping Location 3
Location
Yellow light duration (sec): 8.525
Signal phase number: 5

Stopping Location 4
Location
Yellow light duration (sec): 12.250
Signal phase number: 6

OK Cancel

Figure 6-8. Traffic Signal Information Screen Layout

Location Information

Latitude
Degrees
33.333330

Longitude
Degrees
-117.1234560

Altitude (meters)
100.25

Direction (Degrees)
23.5

OK Cancel

Figure 6-9. Location Information Screen Dialog Box

Table 6-3. Traffic Signal Information Screen Field Definitions

Group	Field	Description
Intersection	Number of Lanes	Set to the number of lanes in the intersection.
Stopping locations 1-4	Yellow light duration	Set to the duration of the yellow light.
	Signal phase number	Set to a number in the range of 1 to 8. Each stopping location should have a different signal phase number.

Group	Field	Description
Location Information Dialog Box	Latitude	Set to the latitude of the intersection or stopping location. Enter northern latitudes as positive numbers and southern latitudes as negative numbers.
	Longitude	Set to the longitude of the intersection or stopping location. Enter northern longitudes as positive numbers and southern longitudes as negative numbers..
	Altitude	Set to the altitude (height above ellipsoid).
	Direction	Set to the direction. 0 represents north. Enter eastern directions as positive numbers and western directions as negative numbers.

6.6.4.7 Test Display Screen

The Test Display Screen provides a real time display of the WAVE network status. The Test Display screen also enables the user to pause and resume a test, or to quit a test. Figure 6-10 shows the Test Display Screen layout and Table 6-4 provides the field definitions.

The screenshot shows a software window titled "Test Display". It contains two main data tables, one for OBUs and one for RSUs, each with 5 columns. The OBU table headers are Sender ID, Msg Count, Distance (m), Rel Heading, RSSI, and Speed (m/s). The RSU table headers are Sender ID, Msg Count, Distance (m), Bearing, RSSI (dBm), Latitude, and Longitude. Below these tables is a status section with "Status" (Log File ID: N/A, Messages Sent: 78, Elapsed Seconds: 8 of 12000) and "Local Status" (Sender ID: 103, Latitude: 0.000000, Longitude: 0.000000, Speed (m/s): 0.000). At the bottom left are "Pause" and "Quit" buttons.

OBU	1	2	3	4	5
Sender ID					
Msg Count					
Distance (m)					
Rel Heading					
RSSI					
Speed (m/s)					

RSU	1	2	3	4	5
Sender ID					
Msg Count					
Distance (m)					
Bearing					
RSSI (dBm)					
Latitude					
Longitude					

Status Log File ID: N/A Messages Sent: 78 Elapsed Seconds: 8 of 12000		Local Status Sender ID: 103 Latitude: 0.000000 Longitude: 0.000000 Speed (m/s): 0.000
---	--	--

Pause Quit

Figure 6-10. Test Display Screen Layout

Table 6-4. Test Display Screen Field Definitions

Group	Field	Description
OBU	Sender ID	Displays the sender ID of the OBU.
	Msg Count	Displays the number of messages received from the OBU since the beginning of the test.
	Distance	Displays the distance to the OBU.
	Rel Heading	Displays the relative heading to the OBU (i.e., bearing from the local OBU to the remote OBU minus the remote OBU's heading) if the local unit is an OBU. Displays the remote OBU's heading if the local unit is an RSU.
	RSSI	Displays the received signal strength of the last message received from the OBU.
	Speed	Displays the speed of the OBU obtained from the CAN bus.
RSU	Sender ID	Displays the sender ID of the RSU.
	Msg Count	Displays the number of messages received from the RSU since the beginning of the test.
	Distance	Displays the distance to the RSU.
	Bearing	Displays the bearing to the RSU.
	RSSI	Displays the received signal strength of the last message received from the RSU. The T9App automatically adjusts the RSSI by a known offset prior to display or logging, based on the WRM IP address. DENSO measured this offset prior to shipment. If the IP address is changed, an incorrect offset or no offset may be applied.
	Latitude	Displays the latitude of the RSU.
	Longitude	Displays the longitude of the RSU.
Control	Pause Button	Pauses the current testing. The T9App stops the screen updates log generation, and changes the button label to "Resume".
	Resume Button	Resumes a paused test. The T9App resumes the screen updates and log generation, and changes the button label to "Pause".
	Quit Button	Quits the current test. The T9App stops the screen updates, closes the logfile, and exits to the main screen. The logfile contains the log entries generated up to the time the test was stopped.

Group	Field	Description
Status	Log File ID	Displays the current test ID that will be used in the log file name. The T9App sets this number to 1 for the first logged run of the day and automatically increments it during the day. The T9App resets the test number to 1 if the test name or log directory is changed on the Test Options Screen (see Section 6.6.4.5). The T9App does not increment the test number for tests that are not logged.
	Messages Sent	Displays the number of messages sent since the beginning of the test. If the Test Control selection on the Test Options Screen (see Section 6.6.4.5) is "Run for n sent messages", the T9App displays Messages Sent as "x of n".
	Seconds Elapsed	Displays the elapsed time since the beginning of the test. This does not include any time during which the test was paused. If the Test Control selection on the Test Options Screen (see Section 6.6.4.5) is "Run for n seconds", the T9App displays Seconds Elapsed as "x of n".
Local	Sender ID	Displays the Sender ID of the local unit.
	Latitude	Displays the latitude of the local unit.
	Longitude	Displays the longitude of the local unit.
	Speed	Displays the speed of the local unit obtained from the CAN bus (applicable for OBUs only).

6.6.4.8 Logfile Usage

Logfile Contents

If the user requests logging on the Test Options Screen (see Section 6.6.4.5), the T9App generates a logfile with configuration information and entries for each message sent and received. Table 6-5. Logfile Record Types describes each of the log record types, and indicates whether or not they appear in an OBU or RSU log.

Table 6-5. Logfile Record Types

Record Type	Contents	OBU Log	RSU Log
CONFIG	Records the configuration data entered by the user on the T9App GUI.	√	√
OBUTX	Records data sent over the air by the local unit when the unit is operating in OBU mode.	√	
OBURX	Records data received from a remote OBU.	√	√
RSUTX	Records data sent over the air by the local unit when the unit is operating in RSU mode.		√
RSURX	Records data received from a remote RSU.	√	√

The T9App writes a heading record before the first record of each type. The heading record contains the names of all of the fields for the record type. The T9App writes a CONFIG record at the beginning of the test. When operating as an OBU, it writes an OBUTX record whenever it transmits a message. When operating as a RSU, it writes a RSUTX record whenever it transmits a message. All units log all received messages as OBURX or RSURX records, depending on the type of the sending unit.

Figure 6-11 shows the contents of a sample OBU logfile. The first two rows are a CONFIG heading record followed by a CONFIG data record. The third row is an OBUTX heading record, followed by seven OBUTX data records. Next is an RSURX heading record followed by two RSURX data records.

```

CONFIG,Message_Size,Message_Rate,Sender_Id,Dest_Ip_Address,Dest_MAC_Address,Traf
CONFIG,71,100,110,192.168.1.255,ff:ff:ff:ff:ff:ff,100,0,0,1,1,0,0,0,1,0,0,TestLo
OBUTX,Length,GPS_Sec_in_Wk,Hdr_Longitude,Hdr_Latitude,Hdr_Altitude,Hdr_Heading,S
OBUTX,71,0,0.0000000,0.0000000,0.,0.,110,0,1,20,00447bcd3de3ae,0,12:c9:19:5e:82:
OBUTX,71,0,0.0000000,0.0000000,0.,0.,110,1,1,20,00447bd13f787d,0,12:c9:19:5e:82:
OBUTX,71,0,0.0000000,0.0000000,0.,0.,110,2,1,20,00447bd7341a3b,0,12:c9:19:5e:82:
OBUTX,71,0,0.0000000,0.0000000,0.,0.,110,3,1,20,00447bdd2af083,0,12:c9:19:5e:82:
OBUTX,71,0,0.0000000,0.0000000,0.,0.,110,4,1,20,00447be31e6e2d,0,12:c9:19:5e:82:
OBUTX,71,0,0.0000000,0.0000000,0.,0.,110,5,1,20,00447be913f5e0,0,12:c9:19:5e:82:
OBUTX,71,0,0.0000000,0.0000000,0.,0.,110,6,1,20,00447bef0c4310,0,12:c9:19:5e:82:
RSURX,Length,GPS_Sec_in_Wk,Hdr_Longitude,Hdr_Latitude,Hdr_Altitude,Hdr_Heading,S
RSURX,125,0,180.0000000,90.0000000,-1000.,0.,104,0,1,20,005957d6528957,1,86:d9:f
RSURX,125,0,180.0000000,90.0000000,-1000.,0.,104,1,1,20,005957da2bf8ec,1,86:d9:f
OBUTX,71,0,0.0000000,0.0000000,0.,0.,110,7,1,20,00447bf50307e8,0,12:c9:19:5e:82:
RSURX,125,0,180.0000000,90.0000000,-1000.,0.,104,2,1,20,005957e0229115,1,86:d9:f
OBUTX,71,0,-117.2275200,33.1333028,143.4,0.,110,8,1,20,00447bfaf8e8f9,0,12:c9:19:
RSURX,125,0,180.0000000,90.0000000,-1000.,0.,104,3,1,20,005957e6195ea5,1,86:d9:f
OBUTX,71,0,-117.2275200,33.1333028,143.4,-142.25,110,9,1,20,00447c00ee8ad4,0,12:
OBUTX,71,470199,-117.2275200,33.1333028,143.4,-142.25,110,10,1,20,00447c06ef2308
RSURX,125,0,180.0000000,90.0000000,-1000.,0.,104,4,1,20,005957ec0e5e1d,1,86:d9:f
RSURX,125,0,180.0000000,90.0000000,-1000.,0.,104,5,1,20,005957f2042ca7,1,86:d9:f
OBUTX,71,470199,-117.2275200,33.1333028,143.4,-142.25,110,11,1,20,00447c0cdfb56
RSURX,125,0,180.0000000,90.0000000,-1000.,0.,104,6,1,20,005957f7fa2b25,1,86:d9:f
OBUTX,71,470199,-117.2275200,33.1333028,143.4,-142.25,110,12,1,20,00447c12cff9b7
OBUTX,71,470199,-117.2275200,33.1333028,143.4,-142.25,110,13,1,20,00447c18c69ddd
RSURX,125,0,180.0000000,90.0000000,-1000.,0.,104,7,1,20,005957fe1ed7c6,1,86:d9:f

```

Figure 6-11. Sample OBU Logfile

6.6.4.9 Importing a Logfile into Excel

One method of converting the logfile into a more easily readable format is to import it into Excel. The procedure for this is as follows:

1. Open Excel.
2. Use the open file command and select the desired logfile. Excel will display the Text Import Wizard, step 1 of 3. Leave the default settings, select Next and step 2 of 3 will be displayed.
3. On step 2, check the box to indicate comma is a delimiter and then select Finish. See Figure 6-12.

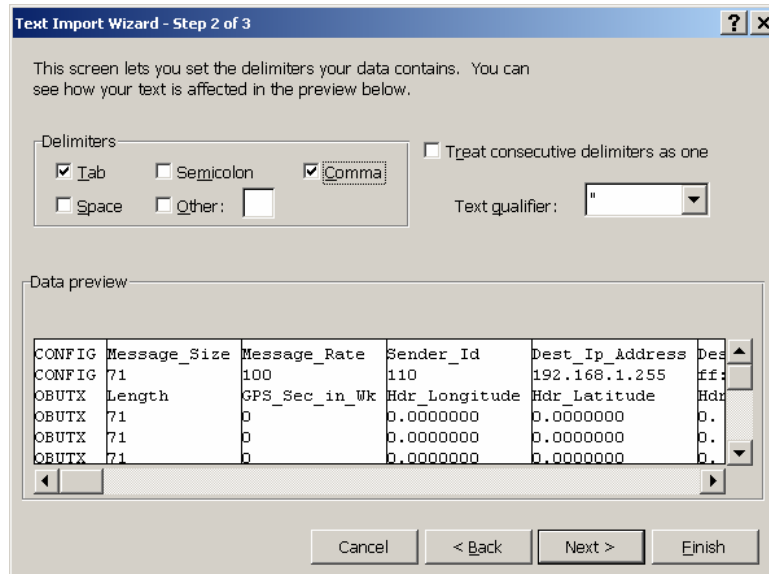


Figure 6-12. Excel Text Import Wizard

4. Excel will now display the logfile in a worksheet. Push the select all button (above the "1" and to the left of "A"). Excel will highlight all of the cells as shown in Figure 6-13.

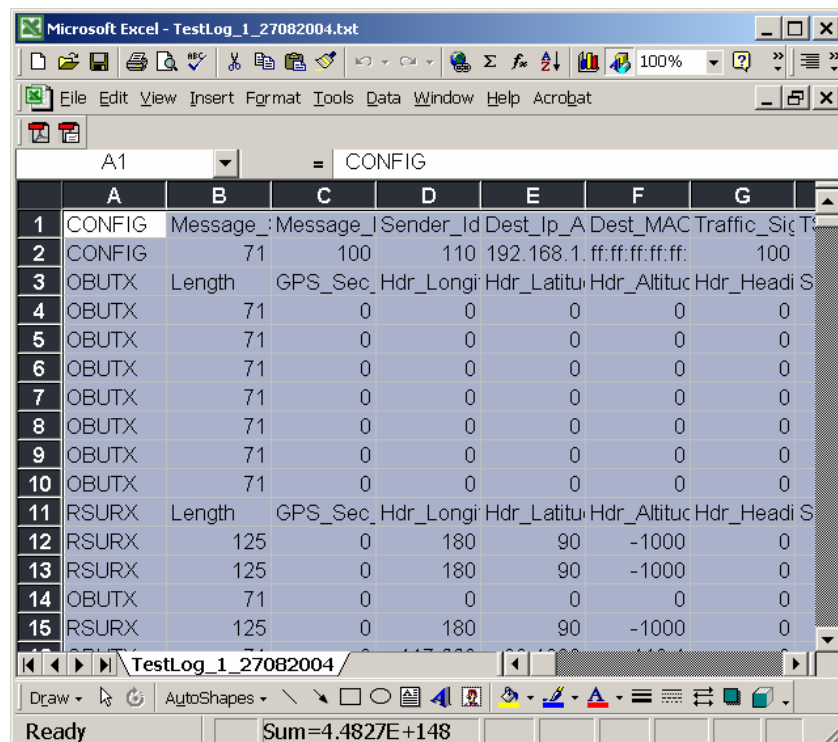


Figure 6-13. Worksheet Selection

- From the menu, select Format, Column, AutoFit Selection. The logfile will now be displayed in a more easily read format. See Figure 6-14.

	A	B	C	D	E
1	CONFIG	Message_Size	Message_Rate	Sender_Id	Dest_Ip_Address
2	CONFIG	71	100	110	192.168.1.255
3	OBUTX	Length	GPS_Sec_in_Wk	Hdr_Longitude	Hdr_Latitude
4	OBUTX	71	0	0	0
5	OBUTX	71	0	0	0
6	OBUTX	71	0	0	0
7	OBUTX	71	0	0	0
8	OBUTX	71	0	0	0
9	OBUTX	71	0	0	0
10	OBUTX	71	0	0	0
11	RSURX	Length	GPS_Sec_in_Wk	Hdr_Longitude	Hdr_Latitude
12	RSURX	125	0	180	90
13	RSURX	125	0	180	90
14	OBUTX	71	0	0	0
15	RSURX	125	0	180	90

Figure 6-14. Imported Logfile

6.6.5 Troubleshooting

This chapter describes the error messages generated by the T9App and how to resolve them.

6.6.5.1 WRM Communication

The T9App attempts to retrieve the current WRM configuration upon startup. If it is successful, it displays the message "WRM Config loaded OK" at the bottom of the main screen. If it is unable to retrieve the configuration, it displays the message "Unable to init WRM API". If this message appears, check the following.

- Verify the HD and WRM Ethernet interfaces are connected with the correct cabling.
- Verify the WRM is powered on.
- Verify the WRM is in WAVE mode. Use a Telnet command to change the mode from 802.11a to WAVE if necessary.

6.6.5.2 GPS and Traffic Signal Communication

The GPS and Traffic Signal serial port configurations are set using the Comm Parameters Screen (see Section 6.6.4.4). If both are enabled, they must use different COM ports. If the same port is selected for both devices, the T9App displays the message box shown in Figure 6-16 when the user attempts to exit the screen. To eliminate the error, select different COM ports for each device, or disable the setup for one of the devices.

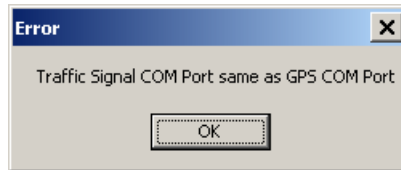


Figure 6-15. COM Port Conflict Error Message

6.6.5.3 GPS Communication

The T9App attempts to initialize the selected GPS serial port at the beginning of each test when GPS setup is enabled on the Comm Parameters screen. If it is unable to initialize the port, it displays the message box shown in Figure 6-16.

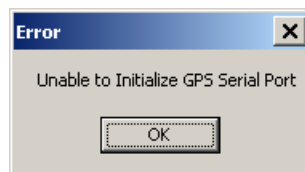


Figure 6-16. GPS Port Initialization Error Message

If this message box appears, check the following.

1. Verify the selected COM port exists on the host device (check the number of ports available using the OS).
2. Verify the GPS is connected to the selected port.
3. Verify no other application is using the COM port (e.g., Hyperterm).
4. Reboot the machine if the above steps do not resolve the issue. The COM port may be hung from use by a previous application.

If the T9App does not appear to be receiving GPS data, verify the serial port settings on the GPS receiver match the T9App settings.

6.6.5.4 Traffic Signal Communication

The T9App attempts to initialize the selected Traffic Signal port at the beginning of each test when operating in RSU mode and Traffic Signal setup is enabled on

the Comm Parameters Screen. If it is unable to initialize the port, it displays the message box shown in Figure 6-17.

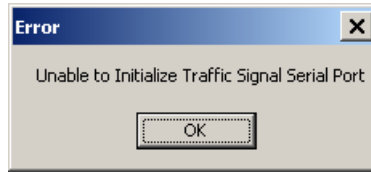


Figure 6-17. Traffic Signal Port Initialization Error Message

If this message box appears, check the following.

1. Verify the selected COM port exists on the host device (check the number of ports available using the OS).
2. Verify the Traffic Signal is connected to the selected port.
3. Verify no other application is using the COM port (e.g., HyperTerminal).
4. Reboot the machine if the above steps do not resolve the issue. The COM port may be hung from use by a previous application.

If the T9App does not appear to be receiving Traffic Signal data, verify the serial port settings on the Traffic Signal controller match the T9App settings.

6.6.5.5 CAN Bus Communication

The T9App attempts to initialize the CAN bus when operating in OBU mode and CAN setup is enabled on the Comm Parameters Screen. If it is unable to initialize the bus, it displays the message box shown in Figure 6-18. Select OK to continue the test without using the CAN interface. This is equivalent to unchecking the CAN setup box on the Comm Parameters Screen (see Section 6.6.4.4) Select Cancel to return to the main screen without executing the test.

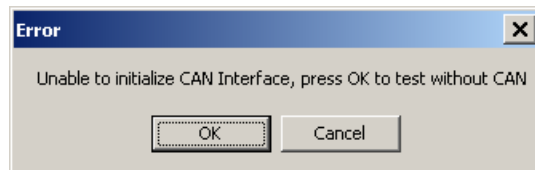


Figure 6-18. CAN Bus Initialization Error Message Box

If this message box appears and the CAN bus should be operational, check the following.

1. Verify the Grid Connect CAN bus drivers are installed on the host device.
2. Verify the CAN bus settings for baud rate and init type on the Comm Parameters Screen are valid for the current setup.
3. Verify the Grid Connect USB adapter is plugged into the host device.

4. Verify the cable is connected between the Grid Connect adapter and the vehicle.
5. Verify no other CAN application is running on the same machine (e.g., Grid Connect's PCAN View)